

US011563625B1

(12) **United States Patent**
Dixit et al.

(10) **Patent No.:** **US 11,563,625 B1**
(45) **Date of Patent:** **Jan. 24, 2023**

(54) **STATIC AND DYNAMIC
NON-DETERMINISTIC FINITE AUTOMATA
TREE STRUCTURE APPLICATION
APPARATUS AND METHOD**

(71) Applicant: **RAKUTEN MOBILE, INC.**, Tokyo
(JP)

(72) Inventors: **Mihirraj Narendra Dixit**, Tokyo (JP);
Surender Singh Lamba, Tokyo (JP);
Abhishek Sharma, Tokyo (JP)

(73) Assignee: **RAKUTEN MOBILE, INC.**, Tokyo
(JP)

(*) Notice: Subject to any disclaimer, the term of this
patent is extended or adjusted under 35
U.S.C. 154(b) by 0 days.

(21) Appl. No.: **17/508,912**

(22) Filed: **Oct. 22, 2021**

Related U.S. Application Data

(60) Provisional application No. 63/249,301, filed on Sep.
28, 2021.

(51) **Int. Cl.**
H04L 41/0659 (2022.01)
H04L 41/0677 (2022.01)
(Continued)

(52) **U.S. Cl.**
CPC **H04L 41/0659** (2013.01); **G06N 5/003**
(2013.01); **G06N 5/006** (2013.01);
(Continued)

(58) **Field of Classification Search**
CPC **H04L 41/0659**; **H04L 41/0677**; **H04L**
41/0686; **G06N 5/003**; **G06N 5/006**;
G06N 5/048; **G06N 7/02**
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2008/0271141 A1* 10/2008 Goldman H04L 63/1416
726/22
2009/0265288 A1* 10/2009 Chakravarty G06N 5/022
706/11

(Continued)

OTHER PUBLICATIONS

P. M. Rathod, N. Marathe and A. V. Vidhate, "A survey on Finite Automata based pattern matching techniques for network Intrusion Detection System (NIDS)," 2014 International Conference on Advances in Electronics Computers and Communications, 2014, pp. 1-5, doi: 10.1109/ICAIECC.2014.7002456. (Year: 2014).*

(Continued)

Primary Examiner — Brian Whipple

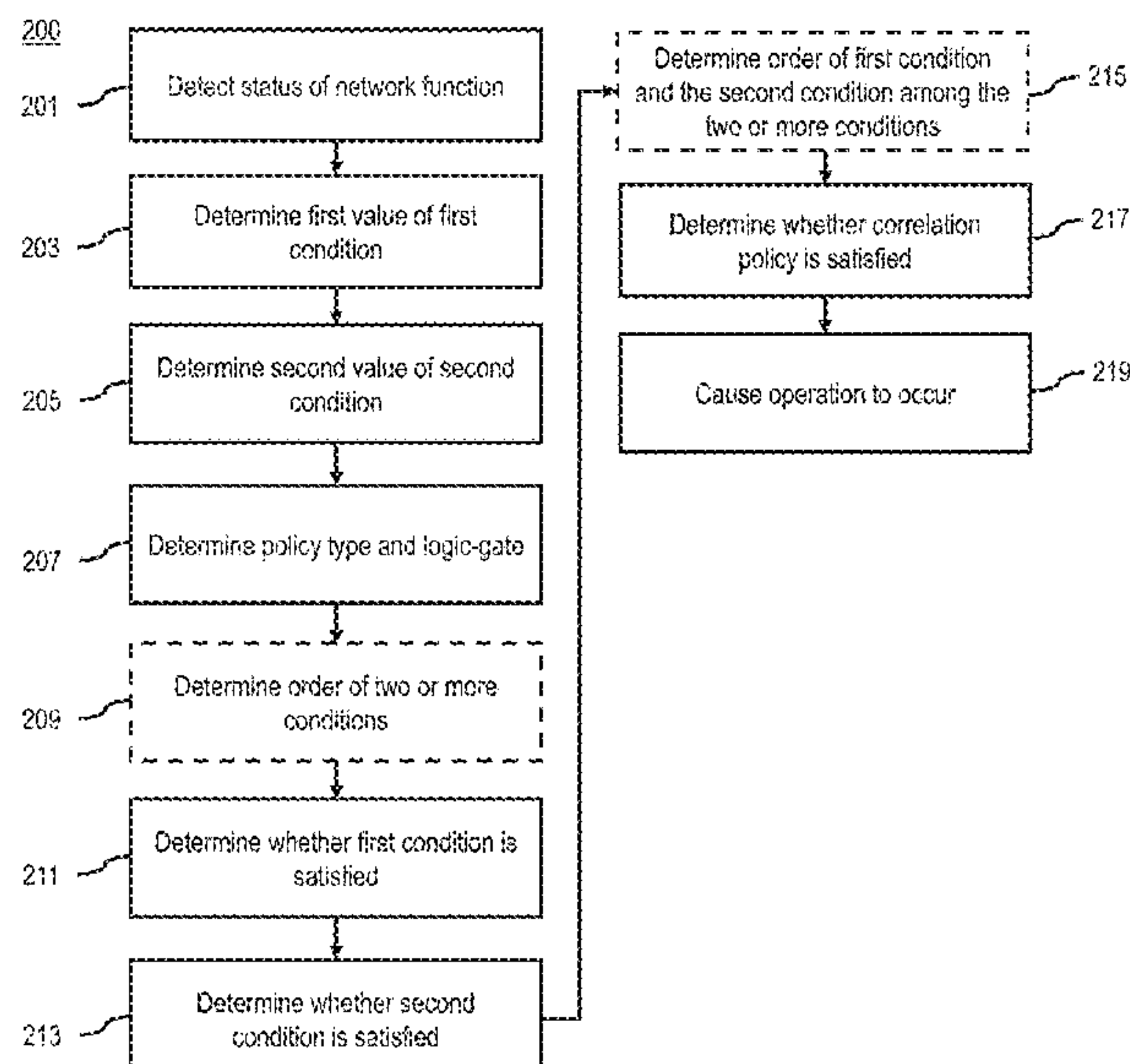
Assistant Examiner — James Ross Hollister

(74) *Attorney, Agent, or Firm* — Hauptman Ham, LLP

(57) **ABSTRACT**

A method includes processing a user input for generating a non-deterministic finite automata tree (NFAT) correlation policy. The user input indicates one or more of a static condition or a dynamic condition for inclusion in the NFAT correlation policy. The static condition includes a comparison between a defined entity and a first fixed parameter. The dynamic condition includes a comparison between the defined entity and a variable parameter. An applicable NFAT element is generated that includes at least one of the NFAT correlation policy generated based on a determination that the user input indicates the static condition or a NFAT template generated based on a determination that the user input indicates the dynamic condition. Event data received from a network device is processed to detect a status of a network entity associated with a communication network based on the applicable NFAT element.

20 Claims, 12 Drawing Sheets



- (51) **Int. Cl.**
G06N 5/00 (2006.01)
H04L 41/0686 (2022.01)
G06N 5/04 (2006.01)
G06N 7/02 (2006.01)
- (52) **U.S. Cl.**
CPC *G06N 5/048* (2013.01); *G06N 7/02*
(2013.01); *H04L 41/0677* (2013.01); *H04L*
41/0686 (2013.01)

(56) **References Cited**

U.S. PATENT DOCUMENTS

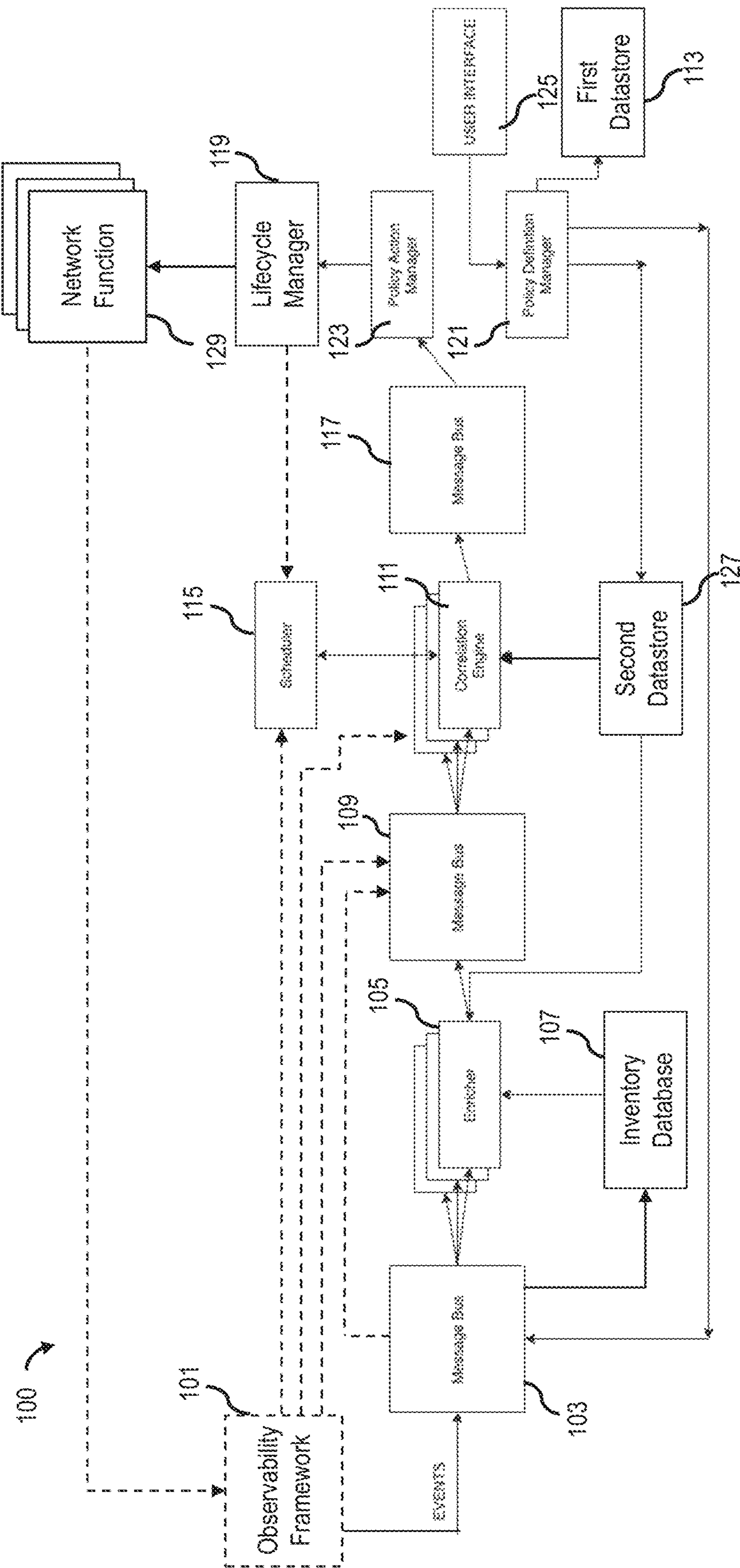
2011/0167030 A1 * 7/2011 Bremler-Barr G06N 5/003
706/48
2019/0228012 A1 * 7/2019 Sadredini G06F 9/4498

OTHER PUBLICATIONS

R. Antonello, S. Fernandes, D. Sadok, J. Kelner and G. Szabo,
“Deterministic Finite Automaton for scalable traffic identification:
The power of compressing by range,” 2012 IEEE Network Opera-
tions and Management Symposium, 2012, pp. 155-162, doi: 10.1109/
NOMS.2012.6211894. (Year: 2012).*

* cited by examiner

FIG. 1



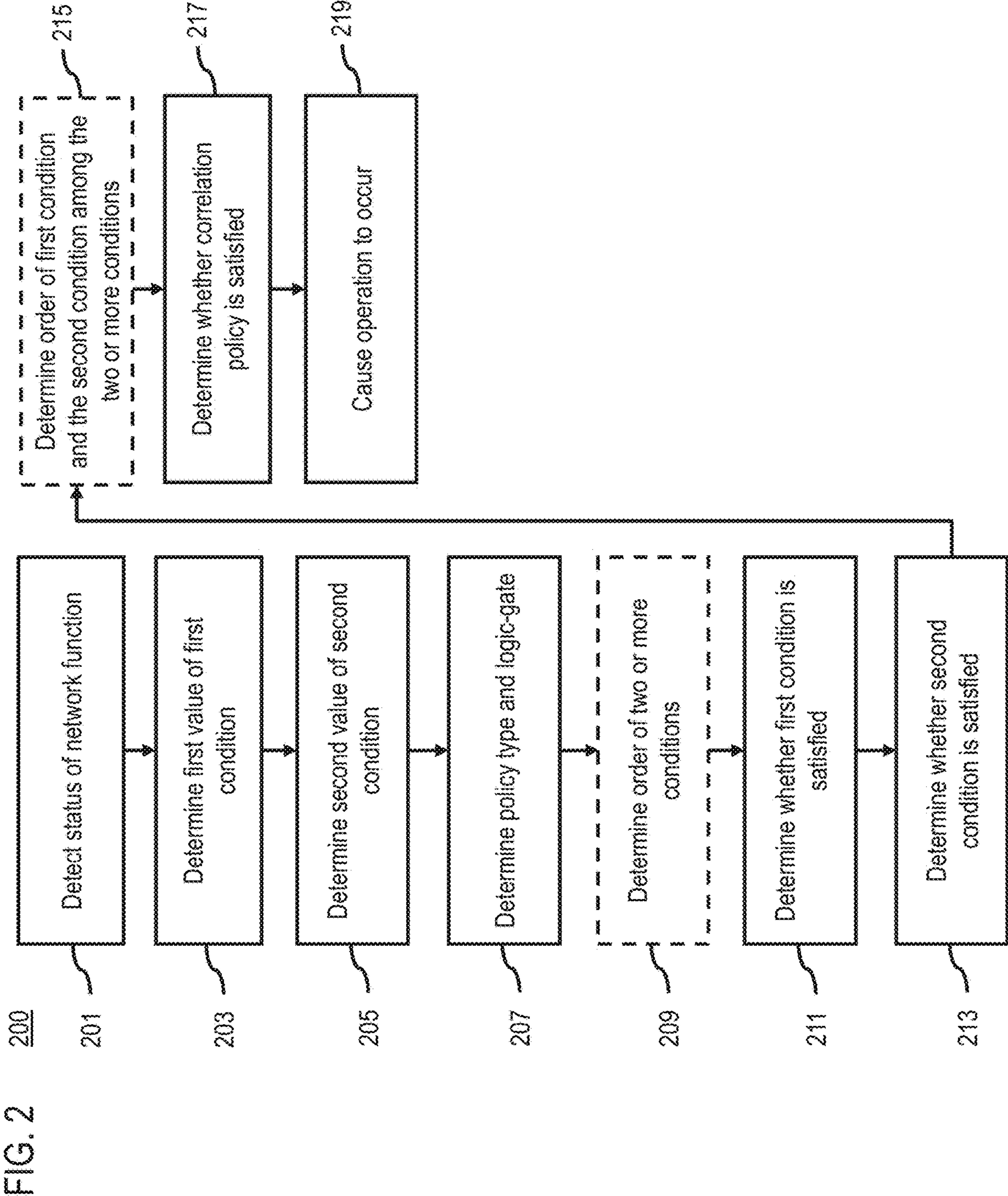


FIG. 3 300

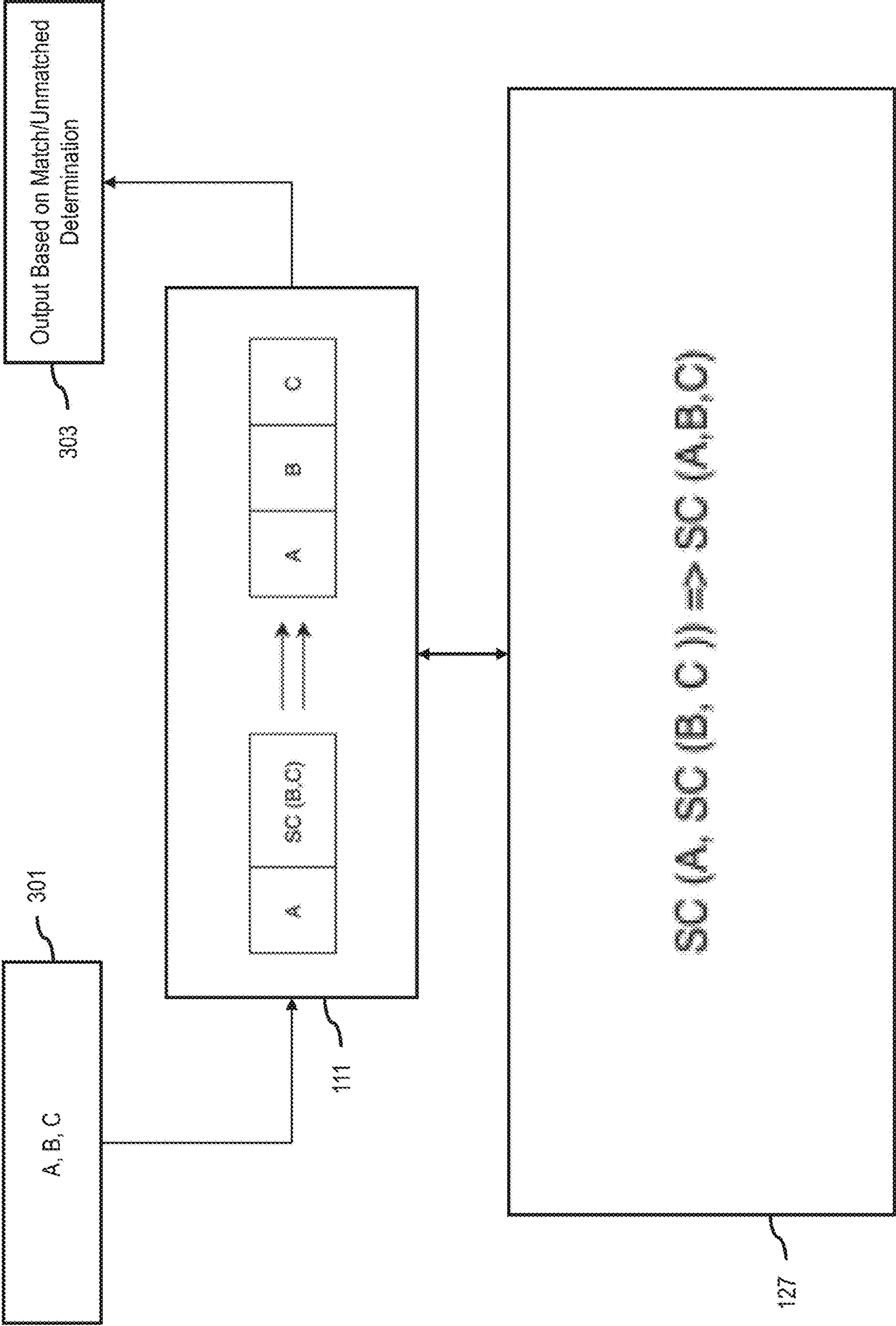


FIG. 4 400

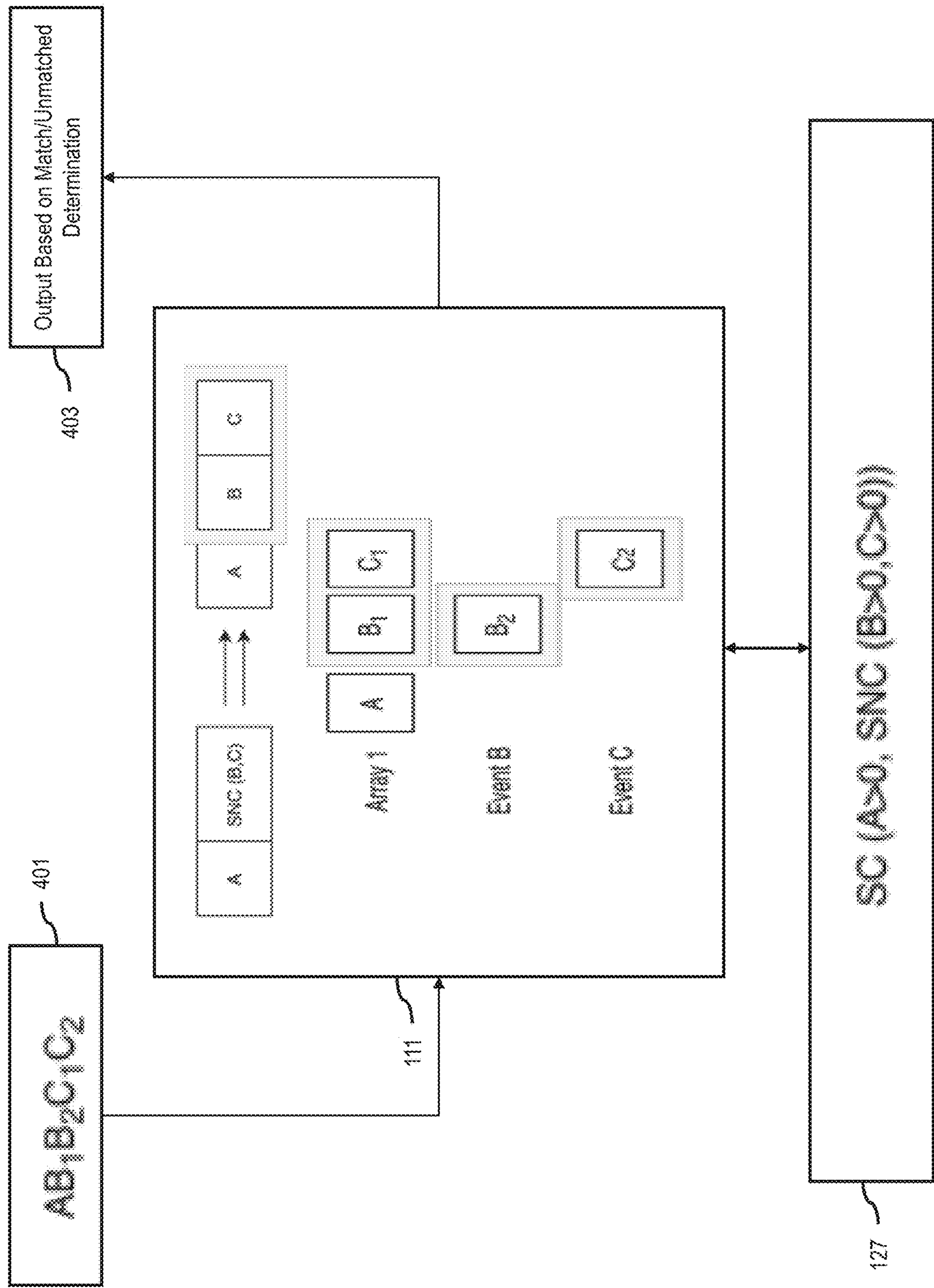


FIG. 5 500

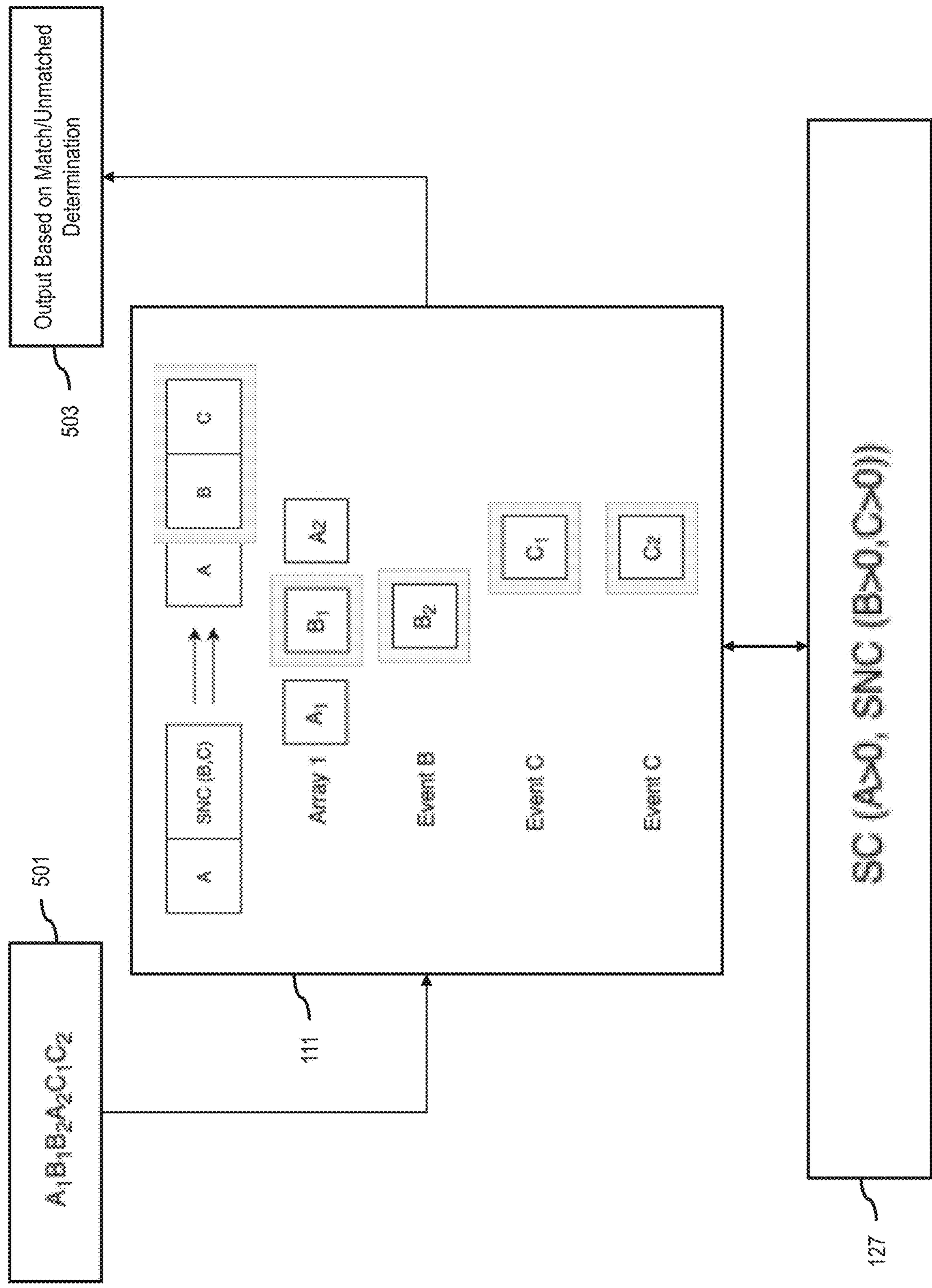


FIG. 6 600

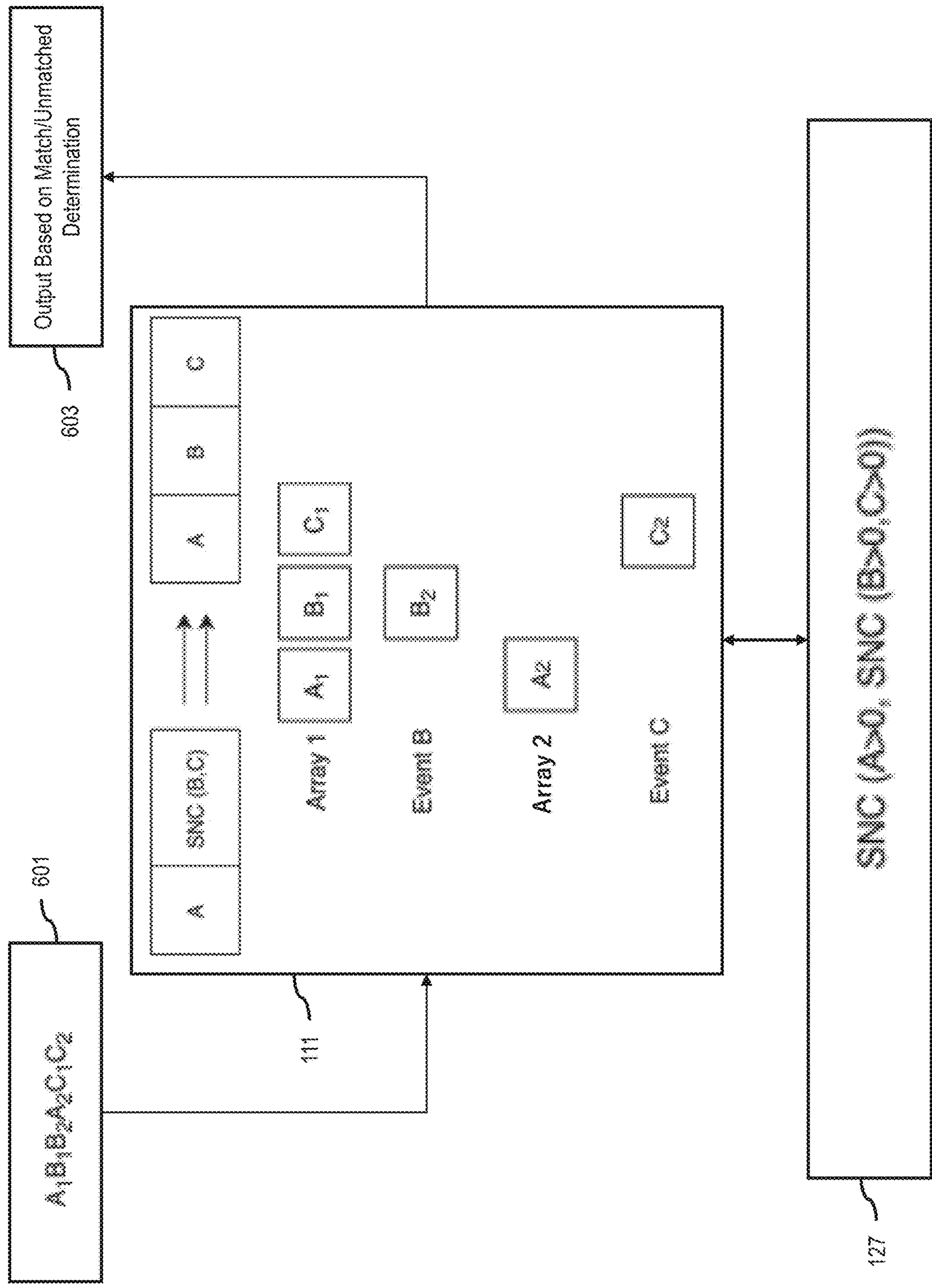


FIG. 7

700

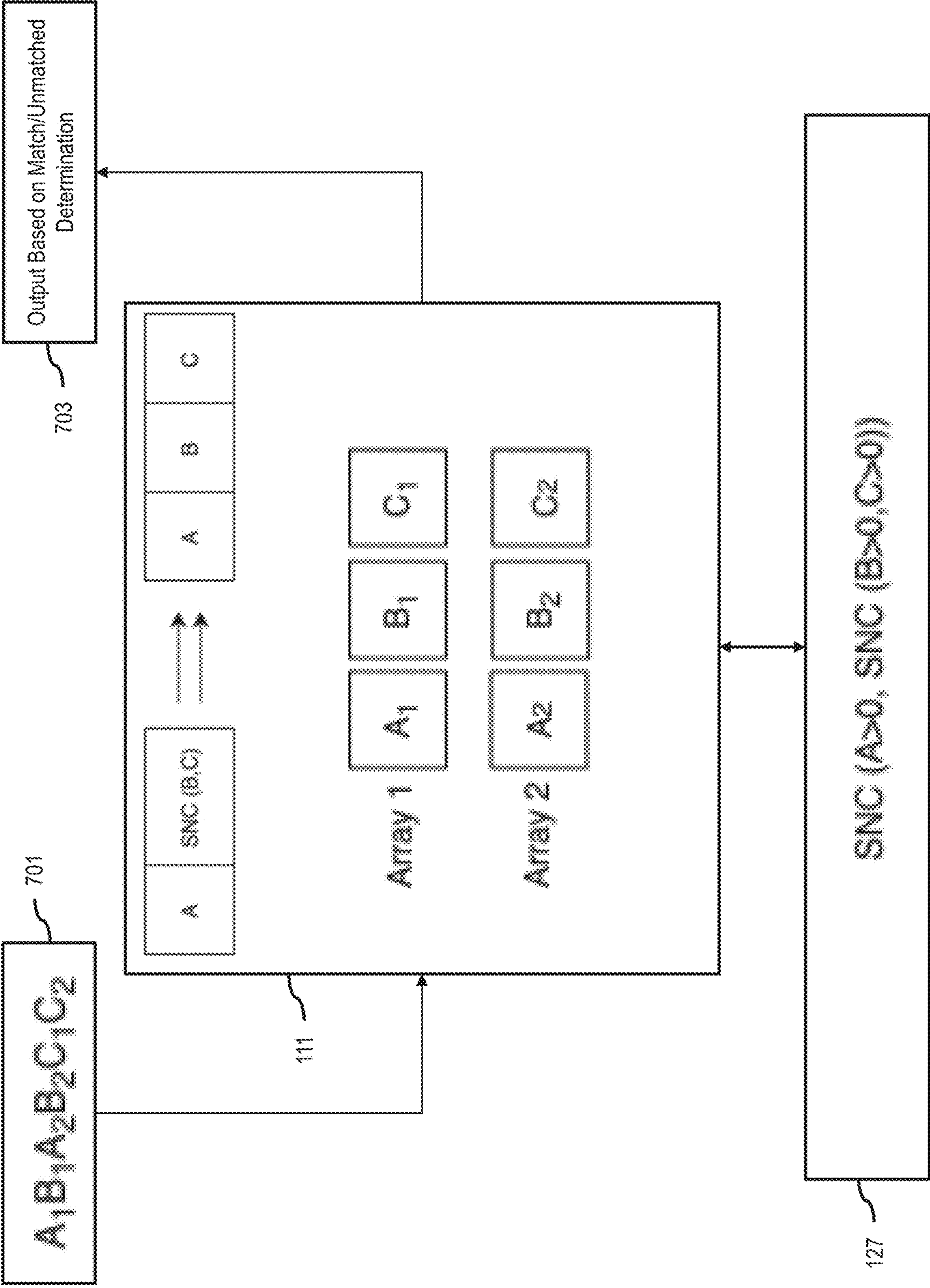


FIG. 8 800

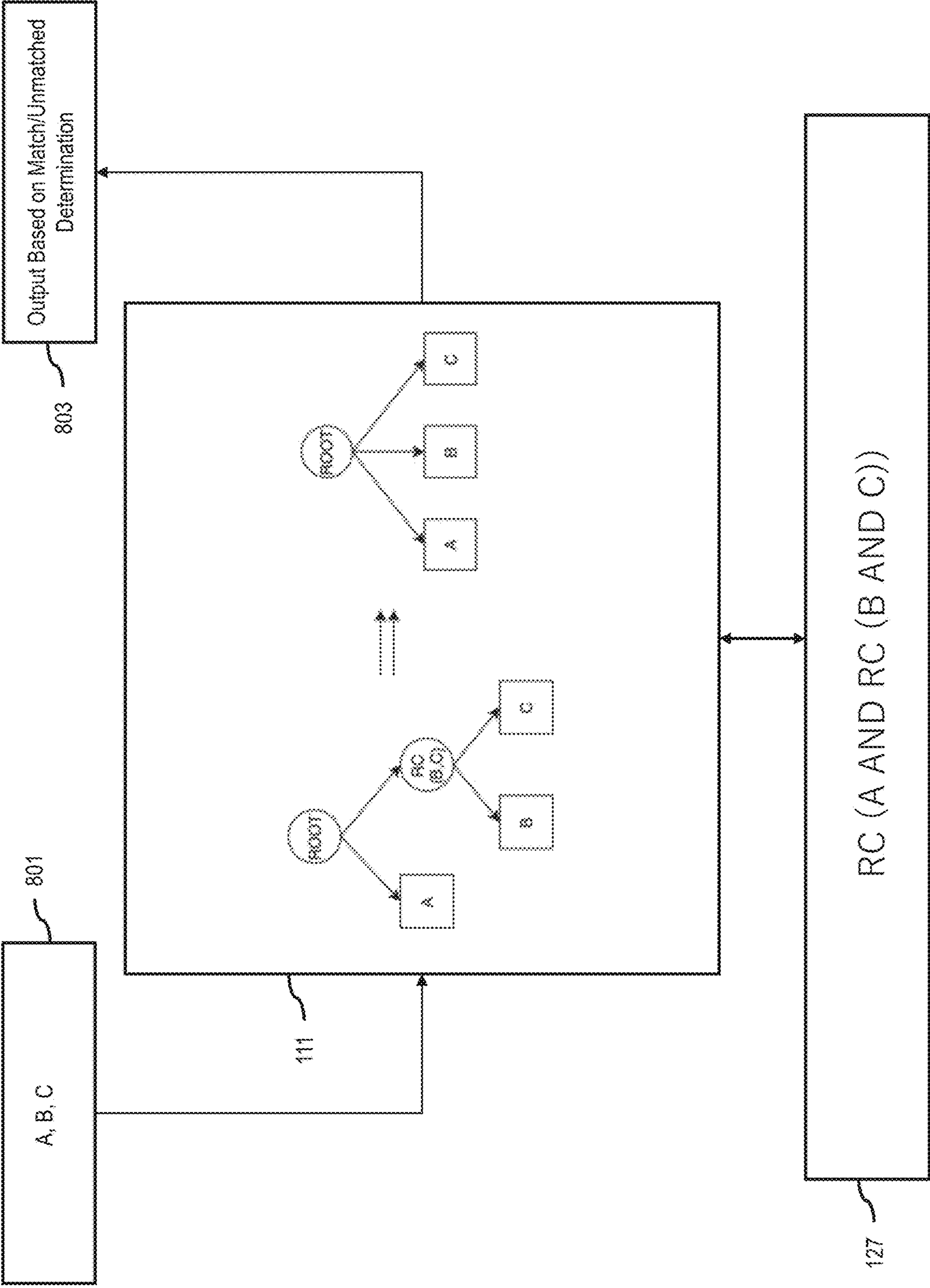


FIG. 9 900

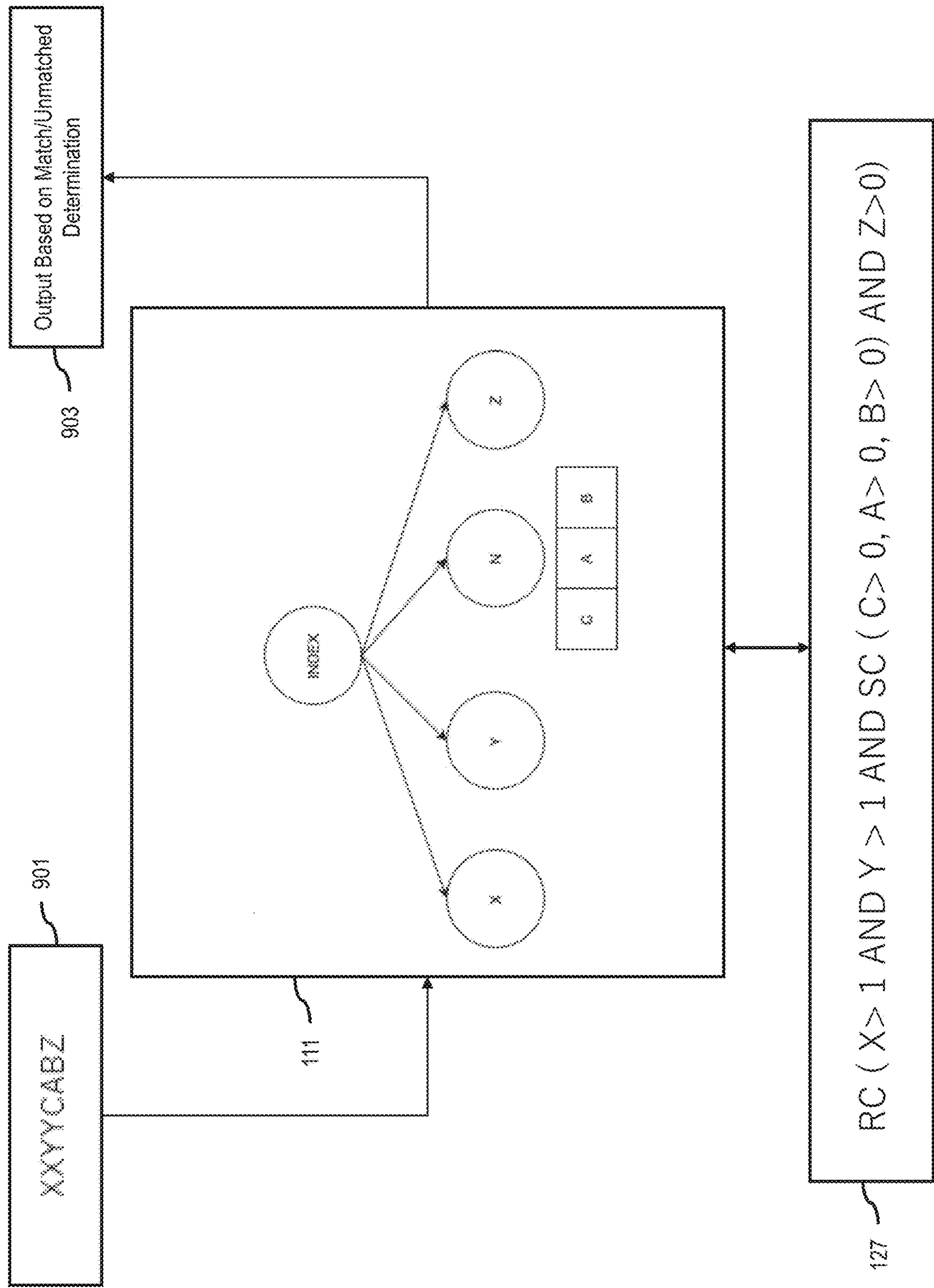


FIG. 10 1000

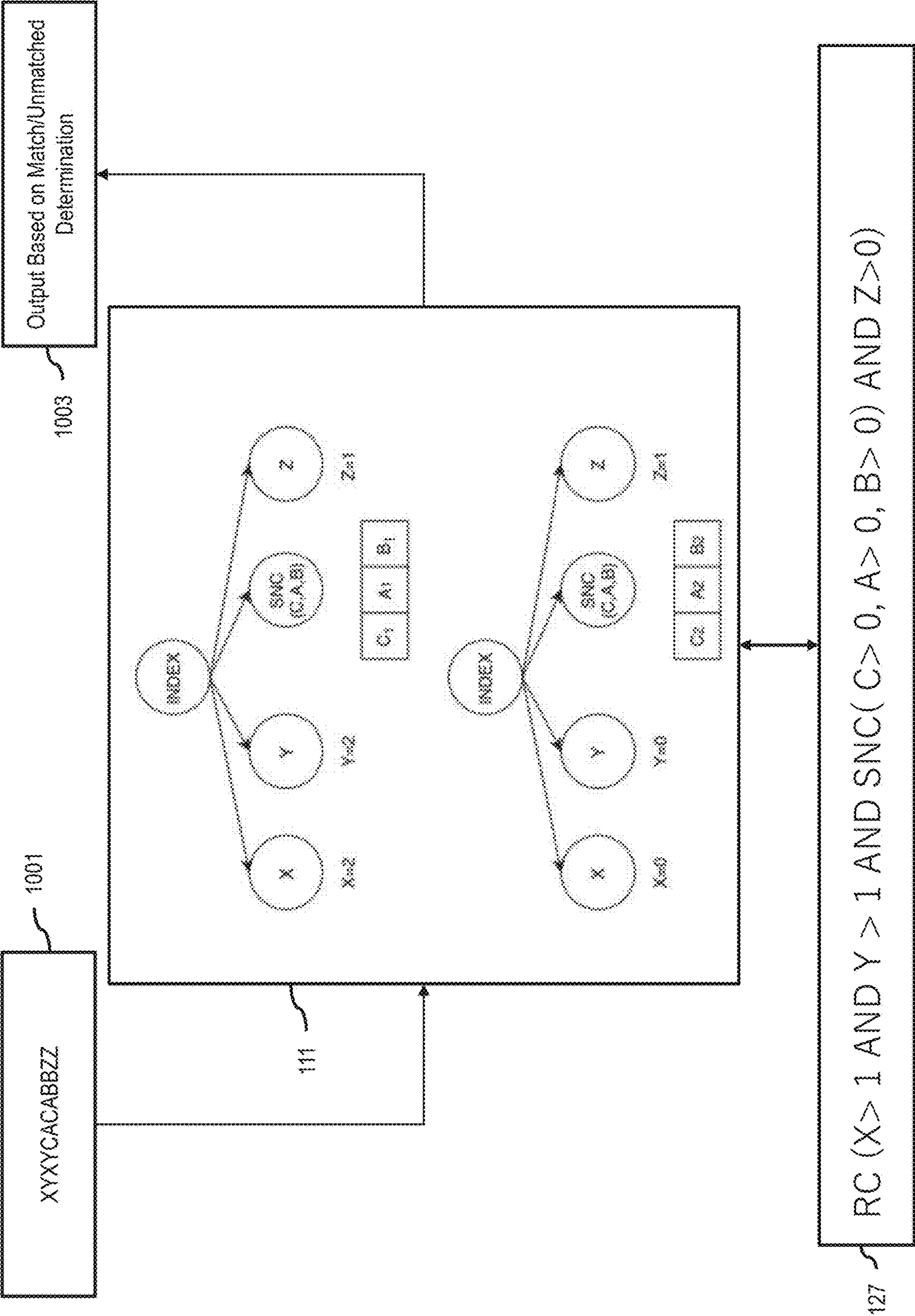


FIG. 11

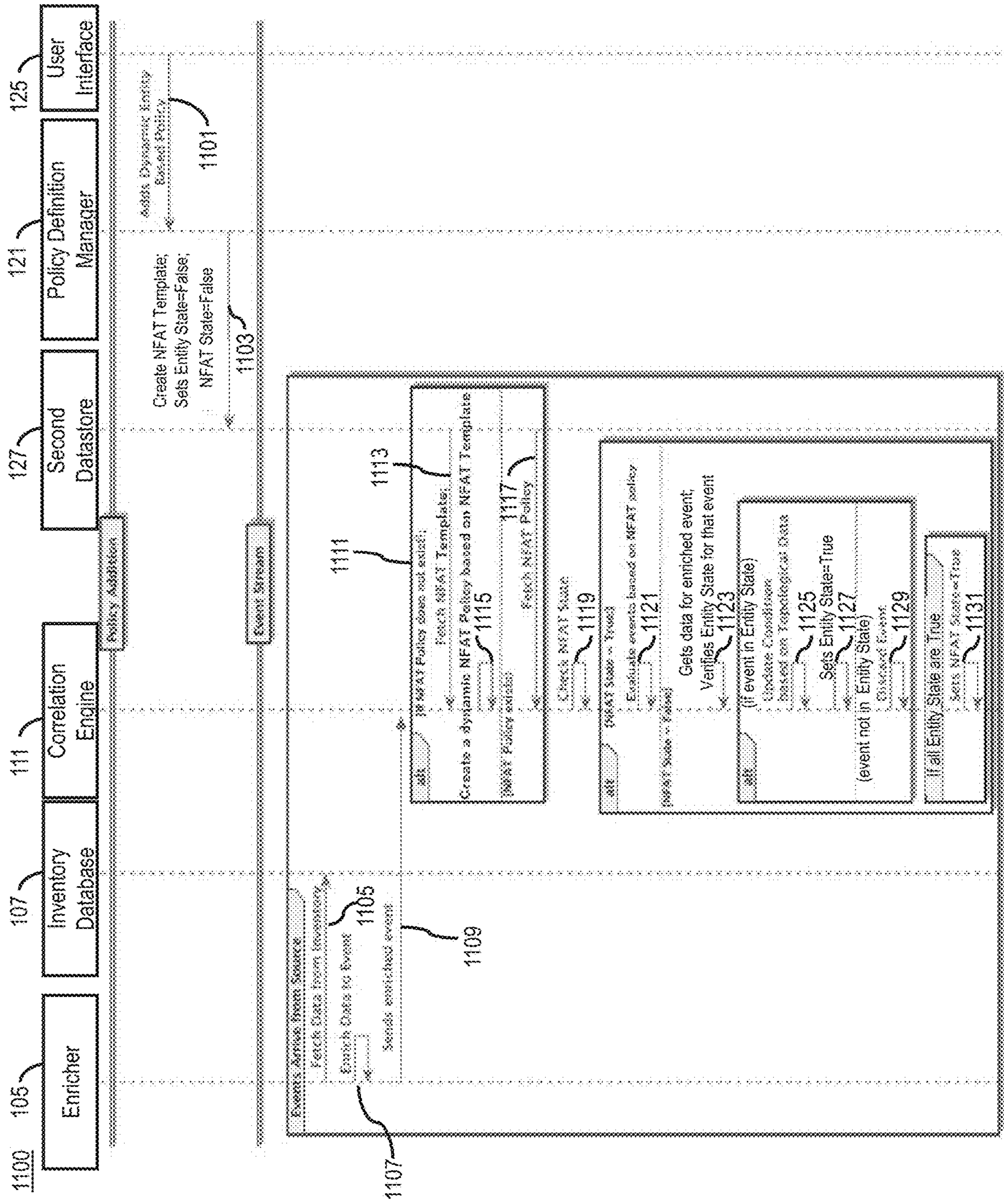
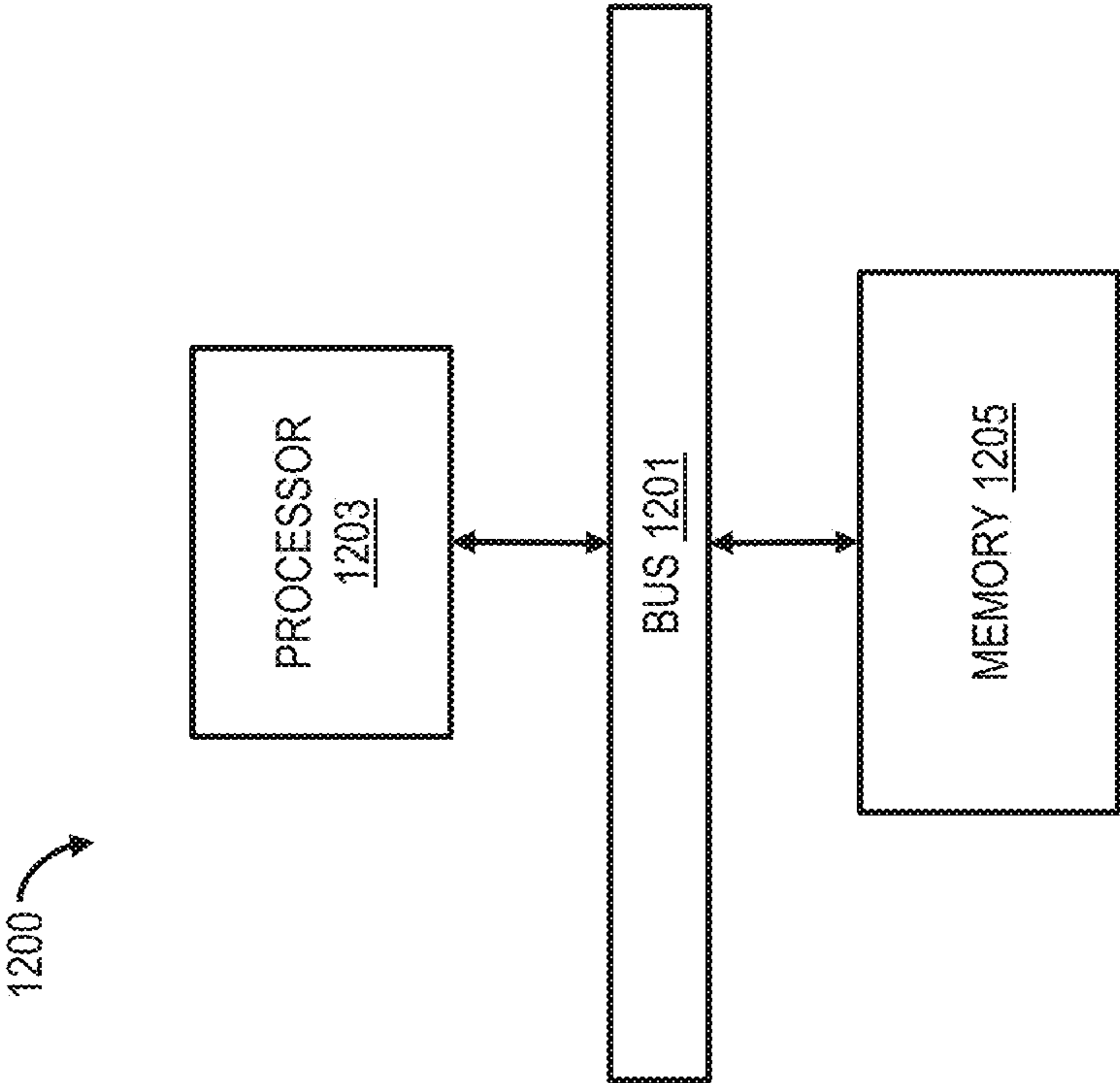


FIG. 12



1

STATIC AND DYNAMIC NON-DETERMINISTIC FINITE AUTOMATA TREE STRUCTURE APPLICATION APPARATUS AND METHOD

PRIORITY CLAIM AND CROSS-REFERENCE

This application claims priority to Provisional Application No. 63/249,301, filed Sep. 28, 2021, which is hereby incorporated by reference in its entirety.

BACKGROUND

Network service providers and device manufacturers (e.g., wireless, cellular, etc.) are continually challenged to deliver value and convenience to consumers by, for example, providing compelling network services that are dependable and capable of being flexibly constructed, scalable and diverse.

BRIEF DESCRIPTION OF DRAWINGS

Aspects of the present disclosure are best understood from the following detailed description when read with the accompanying figures. It is noted that, in accordance with the standard practice in the industry, various features are not drawn to scale. In fact, the dimensions of the various features may be arbitrarily increased or reduced for clarity of discussion.

FIG. 1 is a diagram of a correlation policy application and management system, in accordance with some embodiments.

FIG. 2 is a flowchart of a process for managing and/or applying a logic-gate based NFAT structure correlation policy, in accordance with some embodiments.

FIG. 3 is a block diagram of a logic-gate based NFAT evaluation, in accordance with some embodiments.

FIG. 4 is a block diagram of a logic-gate based NFAT evaluation, in accordance with some embodiments.

FIG. 5 is a block diagram of a logic-gate based NFAT evaluation, in accordance with some embodiments.

FIG. 6 is a block diagram of a logic-gate based NFAT evaluation, in accordance with some embodiments.

FIG. 7 is a block diagram of a logic-gate based NFAT evaluation, in accordance with some embodiments.

FIG. 8 is a block diagram of a logic-gate based NFAT evaluation, in accordance with some embodiments.

FIG. 9 is a block diagram of a logic-gate based NFAT evaluation, in accordance with some embodiments.

FIG. 10 is a block diagram of a logic-gate based NFAT evaluation, in accordance with some embodiments.

FIG. 11 is a sequence diagram of a process for creating, managing and/or applying one or more of a static or a dynamic NFAT structure correlation policy, in accordance with some embodiments.

FIG. 12 is a functional block diagram of a computer or processor-based system upon which or by which some embodiments are implemented.

DETAILED DESCRIPTION

The following disclosure provides many different embodiments, or examples, for implementing different features of the provided subject matter. Specific examples of components and arrangements are described below to simplify the present disclosure. These are, of course, merely examples and are not intended to be limiting. For example,

2

the formation or position of a first feature over or on a second feature in the description that follows may include embodiments in which the first and second features are formed or positioned in direct contact, and may also include embodiments in which additional features may be formed or positioned between the first and second features, such that the first and second features may not be in direct contact. In addition, the present disclosure may repeat reference numerals and/or letters in the various examples. This repetition is for the purpose of simplicity and clarity and does not in itself dictate a relationship between the various embodiments and/or configurations discussed.

Further, spatially relative terms, such as “beneath,” “below,” “lower,” “above,” “upper” and the like, may be used herein for ease of description to describe one element or feature’s relationship to another element(s) or feature(s) as illustrated in the figures. The spatially relative terms are intended to encompass different orientations of an apparatus or object in use or operation in addition to the orientation depicted in the figures. The apparatus may be otherwise oriented (rotated 90 degrees or at other orientations) and the spatially relative descriptors used herein may likewise be interpreted accordingly.

Network services are often provided by inflexible systems that are difficult to configure, scale, and deploy over various target areas. Dependable network systems and/or network services that are capable of being flexibly constructed, scalable and diverse are often dependent on automating the collection, analysis and reporting of information regarding multiple network functions, network services, network devices, etc. that affect the performance, accessibility, configuration, scale, and/or deployment of a communication network, various network functions, network services, and the like.

Conventional network systems are often incapable or fail to efficiently determine the status of a network that involves complex working conditions such as (i) multiple conditions that include conditions for ordered events, conditions for unordered events, and a combination thereof, (ii) conditions for significant quantities of events, or a combination of (i) and (ii).

FIG. 1 is a diagram of a correlation policy application and management system **100**, in accordance with one or more embodiments.

Non-deterministic finite automata tree (NFAT) structure is a type of structure that comprises multiple transition states. The NFAT structure can be an array structure comprising multiple columns, a tree structure comprising multiple leaves, a combination of one or more array structures and one or more tree structures, or some other suitable elements. As will be further discussed in below, evaluation of a NFAT structure allows a system to evaluate any combination of states which cannot be pre-determined, thereby allowing an evaluation of complex network conditions with minimal effort. Specifically, NFAT evaluation executed by system **100** facilitates simultaneous correlation of policies of multiple events or simultaneous correlation of multiple types of policies, thereby enabling the system **100** to monitor the status of a network that involves multiple network functions and complex conditions.

The system **100** makes it possible to collect, analyse, and report information regarding multiple network functions, network services, network devices, etc. that affect the performance, accessibility, configuration, scale, and/or deployment of a communication network, various network functions, network services, and the like. In some embodiments, the system **100** causes one or more operations or actions to

occur based on the analysis and/or reporting of collected information. In some embodiments, the system 100 is configured to observe complex patterns and is capable of specifying if a group of events occur or did not occur after an event.

In some embodiments, the system 100 provides the capability for massively parallel streaming analytics, storage and fast access of data and cached information, even for topology or contextual data enrichment cache timeouts. In some embodiments, the system 100 provides an ability to setup custom functions, rules, and/or timing associated with determining compliance with one or more rules, implementing one or more rules and/or causing one or more processes to occur based on one or more rules. The system 100 is configured to minimize system resource consumption and processing times while increasing a quantity of data streams and correlation policies that are monitored and managed as compared to conventional correlation architectures that facilitate multi-stream creation and/or processing.

As shown in FIG. 1, the system 100 comprises an optional observability framework 101, a first message bus 103, one or more optional data enrichers 105, an inventory database 107, a second message bus 109, one or more correlation engines 111, a first datastore 113, a scheduler 115, a third message bus 117, a lifecycle manager 119, a policy definition manager 121, a policy action manager 123, a user interface 125 and a second datastore 127. It is contemplated that the functions of these components may be combined in one or more components or performed by other components of equivalent functionality. In some embodiments, various components of system 100 individually, or together, are implemented as a set of computer readable instructions that, when executed by a processor such as a processor 1203 (FIG. 12), facilitates the processes discussed with respect to one or more embodiments. In some embodiments, the system 100 is a pluggable correlation architecture.

The observability framework 101 is communicatively coupled with one or more network devices, terminals, nodes, antennas, databases, operating systems, directories, firewalls, detection systems, applications, mainframes, application servers, bare metal monitor, cluster, or other suitable data source capable of generating information that can be processed into event data usable to determine a status of one or more network functions 129. The one or more network functions 129 are associated with operating a communication network. In some embodiments, a network function 129 corresponds to an operation or service performed or provided by one or more hardware components of a communication network, one or more software components of a communication network, or a combination thereof.

The observability framework 101 sends the event data to the first message bus 103. In some embodiments, the observability framework 101 is omitted and the event data is received by message bus 103 or directly input and stored in inventory database 127. The first message bus 103 is communicatively coupled with the policy definition manager 121 and is configured to send the event data received from the observability framework 101 to the one or more data enrichers 105. The first message bus 103 is configured to send the event data to the one or more data enrichers 105 based on a correlation policy. The correlation policy is communicated to the first message bus 103 by the policy definition manager 121. In some embodiments, system 100 is free from including the one or more data enrichers 105 and the first message bus 103. In such a case, the observability framework 101 sends the event data directly to the second message bus 109 if the system 100 includes more than one

correlation engine 111 or to the designated correlation engine 111 based on the correlation policy. In some embodiments, the first message bus 103 is included and the one or more data enrichers 105 are included, and the first message bus 103 bypasses the data enrichers 105 to send the event data to the second message bus 109 or to the correlation engine 111 based on the correlation policy. In some embodiments, the event data can be originated from any suitable sources and be provided, without passing through the observability framework 101, to the first message bus 103, to the data enrichers 105, to the second message bus 109, and/or to the correlation engine 111.

The policy definition manager 121 is connected to the first datastore 113 and to the second datastore 127. The first datastore 113 comprises a plurality of rules associated with monitoring and modifying the one or more network functions 129. The policy definition manager 121 is configured to generate the correlation policy based, at least in part, on one or more of the rules stored in the first datastore 113 and cause the correlation policy to be stored in the second datastore 127.

The one or more optional data enrichers 105 are configured to format the event data based on one or more of at least one template stored in the inventory database 107 or the one or more rules included in the correlation policy stored in the second datastore 127 to generate enriched data. In some embodiments, a template is based on a set of instructions for formatting event data by one or more of aggregating, parsing, normalizing, filtering, adding, or deleting information from the event data to achieve a format defined by the template. In some embodiments, the data enrichers 105 filter event there that is unrelated to events identified by the correlation policy so that the event data received by the correlation engine 111 is free from extraneous event data that is unrelated to the correlation policy.

The second message bus 109 sends the enriched or unenriched data to the one or more correlation engines 111 based on the one or more rules. In some embodiments, the second message bus 109 is excluded from the system 100 and the system 100 is configured such that the first message bus 103, the one or more data enrichers 105, or the observability framework 101 is configured to directly send the enriched or unenriched data to one or more of the correlation engines 111 based on which optional components are included or excluded from system 100.

The one or more correlation engines 111 are configured to process the enriched or unenriched data based on the one or more rules to determine whether the event data satisfies one or more conditions defined by the correlation policy. The one or more correlation engines 111 are also configured to generate one or more outputs indicative of whether the event data satisfies the one or more conditions defined by the correlation policy.

The third message bus 117 is configured to send at least one of the one or more outputs generated by the one or more correlation engines to the policy action manager 123. In some embodiments, the third message bus 117 is excluded from the system 100 and the one or more correlation engines 111 are configured to directly send a generated output to the policy action manager 123.

The policy action manager 123 is communicatively coupled with the lifecycle manager 119 and is configured to trigger an operation to be performed by the lifecycle manager 119. The operation that is performed by the lifecycle manager 119 is associated with monitoring and modifying the one or more network functions 129 based on the at least one output of the one or more outputs. In some embodi-

ments, the operation that is performed by the lifecycle manager **119** is defined by the correlation policy as an action caused to occur based on an outcome of the determination of whether the event data satisfies the one or more conditions defined by the correlation policy and/or whether the correlation policy, as a whole in accordance with one or more logic-gates included in the correlation policy, is satisfied. In some embodiment, the policy action manager **123** can be communicatively coupled to any suitable network element (e.g., correlation policy engine, enricher, policy definition manager, and other elements illustrated in FIG. 1) and is configured to trigger and/or configure an operation to be performed by said network element.

In some embodiments, correlation engine **111** processes event data received from the observability framework **111** to detect a status of a network function **129**. The event data, in this example, is processed based on a correlation policy stored in second data store **127**, wherein the correlation policy defines two or more conditions. In some embodiments, the correlation policy is at least partially defined by a non-deterministic finite automata (NFA) data structure that is evaluated to solve one or more ordered policies, unordered policies, or a combination thereof. In some embodiments, to facilitate dynamic and parallel solving of ordered and/or unordered policies, the correlation policy is at least partially defined by a NFA-Tree (NFAT) data structure that is evaluated.

The NFAT structure correlation policy has a policy type that is one of any predefined policy type that is capable of being processed by the correlation engine **111**. In some embodiments, the predefined policy type is one of a strict contiguous (SC) policy, a strict non-contiguous (SNC) policy, a relaxed contiguous (RC) policy, or a relaxed non-contiguous (RNC) policy. In some embodiments, a NFAT structure correlation policy includes one or more portions that are a combination of two or more different policy types. In some embodiments, a NFAT structure correlation policy includes multiple policies of a same policy type.

For strict and/or relaxed policy types, the correlation engine **111** processes the event data based on criteria defining whether events indicated in an event stream are to occur in a predefined order. Based on these criteria, the correlation engine **111** determines whether an array or a tree structure should be created.

For a strict policy type, the events in the event stream are required to occur in a predefined order. In this situation, correlation engine **111** determines that one or more arrays will be created as the correlation policy is satisfied.

For example, a condition of Strict (A, B) indicates an order of events "A first and B later." When the correlation engine **111** processes the event stream, if the first event in the event stream is A, then the order of "A first and B later" is at least partially fulfilled and thus an array will be created and the number of A in said array will be increased from "0" to "1". If the subsequent event in the event stream is (i) A, in a contiguous situation, the number of A in said array will be increased from "1" to "2"; otherwise, in a non-contiguous situation, a new array will be created and the number of A in the new array will be increased from "0" to "1"; or (ii) B, then the number of B in said array will be increased from "0" to "1".

In some embodiments, the event data is formatted and/or filtered by the one or more data enrichers **105** before being sent to the correlation engine **111** for processing based on the correlation policy to facilitate processing of policy related

events (e.g., event A and event B) only, while preventing non-policy related events from being sent to the correlation engine **111**.

On the other hand, if the first event in the event stream is not A, then the order of "A first and B later" is not fulfilled and no array will be created. If the subsequent event in the event stream is (i) A, then an array will be created and the process as described above will be carried out; or (ii) any event other than A, then no array will be created, and the correlation engine **111** will proceed to process a next event in the event stream if the correlation engine **111** is within a hold time, for example.

For a relaxed policy type, the events indicated in an event stream can occur in any order. Based on these criteria, the correlation engine **111** determines one or more tree structures with corresponding leaf/leaves will be created.

For example, assuming a condition of Relaxed (A, B), if the event stream contains event A and/or event B, the correlation engine **111** will create a tree structure with two leaves (a first leaf for event A and a second leaf for event B). When the correlation engine **111** processes the event stream, if the event stream contains event A, then the number of A in the first leaf will be increased on the order of 1 (e.g., if there are two events A in the event stream, in a contiguous situation, the number of A in the first leaf will be increased to 2; and in a non-contiguous situation, when the condition associated with A is not yet satisfied (e.g., a condition of "A>2"), the number of A in the first leaf will be increased to 2, when the condition associated with A is satisfied (e.g., a condition of "A>0"), a new tree with two leaves will be created and the number of A in the first leaf of the new tree will be increased from "0" to "1"). The correlation engine **111** performs similar processes for event B.

For contiguous/non-contiguous policy types, the correlation engine **111** processes the event data based on criteria defining whether the events indicated in an event stream are to occur adjacent to similar events. Based on this criteria, the correlation engine **111** determines whether the event should be included in one NFAT data structure (e.g., array structure, tree structure, etc.), or whether the correlation engine **111** should create a NFAT data structure for the event.

For a "contiguous" situation, the events in an event stream should be adjacent to similar events. In this situation, when multiple similar events are included in the event stream, the correlation engine **111** groups or includes multiple similar events into a same array/same leaf of one tree structure, so that they are adjacent to each other.

For example, a condition of Contiguous (A, B) indicates that an event A should be adjacent to one or more other events A and/or an event B should be adjacent to one or more other event B. In this regard, if the event stream contains multiple events A, the correlation engine **111** includes multiple events A in a same array or a same leaf of a same tree, and the number of A in the array/the leaf will increase on the order of 1 until an operation is caused to occur by the correlation engine **111** and the array/the tree is deleted by the correlation engine **111**. The correlation engine **111** performs similar processes for event B.

In a "non-contiguous" situation, the events in an event stream are not required to be adjacent to the similar event. Thus, when the event stream contains multiple similar events and the condition defined by the policy is partially matched, the correlation engine **111** creates multiple NFAT data structures.

For example, if the event stream contains 2 events A₁ and A₂, then the correlation engine **111** creates a first NFAT data structure (e.g., array structure, tree structure, etc.) for A₁ and

then creates a second NFAT data structure (e.g., array structure, tree structure, etc.) for A_2 when the condition associated with A_1 in the first NFAT data structure is matched. The correlation engine 111 performs similar processes for event B.

In some embodiments, one or more policies defining Strict/Relaxed conditions can be combined with one or more policies defining Contiguous/Non-Contiguous conditions to form one or more combined policies. For example, said combined policies can be any of a strict contiguous (SC) policy, a strict non-contiguous (SNC) policy, a relaxed contiguous (RC) policy, a relaxed non-contiguous (RNC) policy, or a combination thereof. For example, various policy types can be optionally combined with each other to form a more complex policy, such as a RC of SNC policy, or some other suitable combination of policy types.

The correlation engine 111 determines a first value of a first condition of the two or more conditions, a second value of a second condition of the two or more conditions, and the policy type of the correlation policy. The correlation engine 111 optionally determines an order of the two or more conditions based on a determination that the correlation policy is a policy type for which an order of the two or more conditions matters to determine whether the correlation policy or conditions thereof are satisfied. In some embodiments, the correlation engine 111 always determines the order of the two or more conditions and stores the values of the two or more conditions in the second data store 127 for future use when determining whether the correlation policy or conditions are satisfied, and/or for use when determining whether a different correlation policy or condition is satisfied.

In some embodiments, based on the policy type, the correlation engine 111 determines whether the first value upon being compared to a first preset value in accordance with an operator is indicative of whether the first condition is satisfied, determines whether the second value is upon being compared to a second preset value in accordance with a corresponding operator is indicative of whether the second condition is satisfied, determines whether the correlation policy is satisfied based on the first value, the second value, an optional logic-gate, and the policy type, and causes an operation to occur based on a determination that the correlation policy is satisfied. In some embodiments, the correlation engine 111 further determines an order of the first condition and the second condition among the two or more conditions, and determines whether the correlation policy is satisfied based on the first value, the second value, the order of the first condition and the second condition, the optional logic-gate, and the policy type. In some embodiments, if the policy type is one that is independent of a specified order and/or is independent of time, the correlation engine 111 ignores the order of the conditions in the event data and/or processes the event data without consideration of time. In some embodiments, if the correlation policy is a strict-type policy, the logic-gate is inferred and/or excluded from the determination of whether the correlation policy is satisfied, because a strict-type policy, alone, requires satisfaction of all conditions included in the correlation policy for the correlation policy to be satisfied.

In some embodiments, the correlation engine 111 processes the event data for a predefined hold time until the correlation policy is satisfied or the predefined hold time is expired. In some embodiments, the correlation engine 111 causes the event data and the correlation policy to be discarded from a memory after the predefined hold time based on a determination that the correlation policy is not

satisfied by the end of the predefined hold time. In some embodiments, the correlation engine 111 causes the event data to be discarded after the predefined hold time based on a determination that the correlation policy is satisfied by the end of the predefined hold time, and the correlation engine 111 detects the status of the network function 129 based on new event data and the correlation policy for a subsequent predefined hold time after the predefined hold time. In some embodiments, the predefined hold time is a validity time of the determined first value, and the correlation engine 111 is further caused to determine the first value upon being compared to the first preset value in accordance with the corresponding operator is indicative of whether the first condition is satisfied, process the event data to determine whether the second condition is satisfied during the hold time, and cause the first value to be discarded after the predefined hold time based on a determination that the second condition is not satisfied by the end of the predefined hold time.

In some embodiments, at least one of the NFAT policy and the data structure defined thereof comprises a logic-gate. The logic-gate, for example, indicates that the correlation policy is satisfied if all the conditions included in the correlation policy are satisfied, at least one of the conditions included in the correlation policy is satisfied, none of the conditions included in the correlation policy are satisfied, or some combination thereof. In some embodiments, the logic-gate is an AND-logic-gate, an OR-logic-gate, an XOR-logic-gate, a NOR-logic-gate, a NAND-logic-gate, a XNOR-logic-gate, some other suitable logic-gate, or a combination thereof. A correlation policy that includes an AND-logic-gate between two conditions requires both conditions to be TRUE for the correlation policy to be considered as being satisfied. A correlation policy that includes an OR-logic-gate between two conditions requires at least one of the conditions to be TRUE for the correlation policy to be considered as being satisfied. A correlation policy that includes a XOR-logic-gate between two conditions results in the correlation policy as being satisfied if the overall state of the correlation policy is TRUE. For an XOR-logic-gate, the overall state (i.e., state of a tree structure or an array structure) is TRUE if either one, but not all, of the sub-states (e.g., state of leaves or columns of the NFAT structure) are TRUE, and the overall state is FALSE when all sub-states are FALSE or all sub-states are TRUE. For a NOR-logic-gate, the overall state is TRUE if all sub-states are FALSE, otherwise, the overall state is FALSE. For a NAND-logic-gate the overall state is FALSE if all sub-states are TRUE, otherwise, the overall state is TRUE. For a XNOR-logic-gate, the overall state is TRUE when all sub-states are TRUE or when all sub-states are FALSE; otherwise, the overall state is TRUE. In some embodiments, complex policies include one or more different policy types and/or one more different logic-gates within the correlation policy.

In some embodiments, the correlation engine 111 causes the operation to occur after the end of the predefined hold time based on a determination that the correlation policy is satisfied during the predefined hold time.

In some embodiments, based on a determination that the policy type is a SC policy or a SNC policy, the correlation engine 111 processes the event data until the correlation policy is satisfied. The correlation engine 111 also causes at least a portion of the event data to be discarded based on a determination that the correlation policy cannot be satisfied during the predefined hold time, and/or adjusts the predefined hold time to facilitate processing the event data until the correlation policy is satisfied.

In some embodiments, based on a determination that the policy type is a SC policy or a SNC policy, the correlation engine 111 processes the event data until the correlation policy is satisfied. The correlation engine 111 also causes the event data to be discarded based on a determination that the correlation policy is satisfied, and detects the status of the network function 129 based on new event data and the correlation policy after the correlation policy is determined to be satisfied.

In some embodiments, the correlation policy comprises one or more unordered threshold-based conditions. The determination of whether the event data satisfies the one or more conditions defined by the correlation policy comprises a determination that the event data includes one or more values that meet or exceed a preset threshold value of at least one of the one or more unordered threshold-based conditions. In some embodiments, alternative comparisons to threshold values are included in the correlation policy. For example, although the sign “>” (which indicates “greater than”) is used in the discussed examples for simplicity, one or more embodiments of the correlation policy optionally include other comparison signs like “<” (which indicates “less than”), “>=” (which indicates “greater or equal to”), “=” (which indicates “equal to”), or some other suitable operator.

As a non-limiting example, an unordered threshold-based condition may involve processing event data associated with determining whether a network node is up, down, or not ready. If, for example, the preset threshold values are NodeDown>=1 and NodeNotReady>=0, with a preset hold time of 10 minutes, once the preset threshold value is met or exceeded, there is no need to wait for the entire hold time, and correlation engine 111 causes the policy action manager 123 to trigger an operation to be performed by the lifecycle manager 119.

In some embodiments, the determination that the event data includes one or more values that meet or exceed a preset threshold value of the at least one of the one or more unordered threshold-based conditions is independent from time.

In some embodiments, the policy action manager 123 is configured to trigger the operation to be performed by the lifecycle manager 119 when the correlation engine 111 determines that the event data includes one or more values that meet or exceed the preset threshold value of the at least one of the one or more unordered threshold-based conditions, or an output is received from the correlation engine 111 indicating the same.

In some embodiments, the policy action manager 123 is configured to trigger the operation to be performed by the lifecycle manager 119 within a preset time interval from a moment the correlation engine 111 determines that the event data includes one or more values that meet or exceed the preset threshold value of the at least one of the one or more unordered threshold-based conditions, or an output is received from the correlation engine 111 indicating the same.

As a non-limiting example, an unordered time-based condition may involve processing event data associated with determining whether a particular process associated with a network function was a success or a failure. Such a policy, for example, may involve processes F1SctpFailure and F1SctpSuccess, where preset threshold values for the processes are F1SctpFailure>1 and F1SctpSuccess<1, a preset hold time is set for 10 minutes, and a rule that defines that the policy action manager 123 should wait for an entirety of the hold time before triggering the operation to be performed

by the lifecycle manager 119 when the determined value for F1SctpFailure or F1SctpSuccess is less than or equal to 1.

The scheduler 115 is communicatively coupled with the correlation engine 111. The scheduler 115 is configured to set the preset time interval based on the determination that the event data includes one or more values that meet or exceed the preset threshold value of the at least one of the one or more unordered threshold-based conditions is made.

In some embodiments, the correlation policy comprises an instruction or data indicative of the preset time interval and the scheduler 115 is configured to cause the correlation engine 111 to communicate the preset time interval to the policy action manager 123 with the generated output. In some embodiments, the correlation engine 111 is configured to one or more of send or generate the one or more outputs based on the preset time interval communicated to the correlation engine 111 by the scheduler 115. In some embodiments, the correlation engine 111 is configured to process the enriched data based on a predefined schedule that is communicated to the correlation engine 111 one or more of by way of the scheduler 115 or the correlation policy. In some embodiments, the scheduler 115 is communicatively coupled with the lifecycle manager 119 and with the correlation engine 111, and the scheduler 115 is configured to set the predefined schedule based on feedback received from the lifecycle manager 119. In some embodiments, the scheduler 115 is configured to set the predefined schedule based on feedback received from the observability framework 101. In some embodiments, the scheduler 115 is directly communicatively coupled with the observability framework 101 and configured to set the predefined schedule based on feedback directly received from the observability framework 101. In some embodiments, the scheduler 115 is configured to set the predefined schedule based on information included in enriched data sent to the correlation engine 111. In some embodiments, the scheduler 115 is configured to set the predefined schedule based on feedback received from the lifecycle manager 119 and the observability framework 101. In some embodiments, the system 100 excludes the scheduler 115 and the correlation engine 111 is configured to generate the one or more outputs independent of time or the correlation engine 111 is configured to generate the one or more outputs based on information included in the enriched data.

In some embodiments, the correlation policy comprises a first rule and a second rule, the event data comprises first event data, second event data and third event data. The correlation engine 111 is configured to determine that the event data satisfies the correlation policy based on a determination that the first event data includes a value that, based on a comparison to a first preset value in accordance with a first operator, is indicative that the first condition is satisfied and the second event data includes a value based on a comparison to a second preset value in accordance with a second operator is indicative that the second condition is satisfied. The correlation engine 111 is also configured to determine that the event data satisfies the correlation policy based on a determination that the third event data includes a value that, based on a comparison to a third preset value in accordance with a third operator, is indicative that the third condition is satisfied. The policy action manager 123 is configured to trigger the lifecycle manager 119 to cause an operation corresponding to a third event corresponding to the third event data to occur based on the determination by the correlation engine 111 that the first and second conditions are satisfied. The policy action manager 123 is additionally configured to trigger the lifecycle manager 119 to

11

perform the operation based on the determination that the third condition is satisfied, indicating that the operation has occurred.

In some embodiments, the correlation policy comprises one or more conditions associated with a preset order, the event data comprises two or more data points, and the determination of whether the event data satisfies the one or more conditions defined by the correlation policy comprises a determination that the two or more data points occur in the preset order.

As a non-limiting example, an ordered condition may involve processing event data associated with determining whether a pod failure has occurred followed by a determination that a failed pod has been restarted. If, for example, Event_Pod_Failure is determined to have occurred followed by Event_Pod_Restarted, the order of events matters according to the correlation policy, the condition is determined to be satisfied.

In some embodiments, the policy definition manager **121** is configured to generate at least one of the one or more rules, or the correlation policy, based on a user input received by way of the user interface **125**. The user interface **125** is, for example, capable of being output by way of a display associated with any of a mobile terminal, a fixed terminal, a portable terminal including a desktop computer, laptop computer, notebook computer, netbook computer, tablet computer, wearable circuitry, mobile handset, server, or combination thereof.

In some embodiments, system **100** is configured to be communicatively coupled with, or included as a component of, a network manager or end-to-end orchestrator that controls one or more workflows associated with any network function **129** comprising or associated with one or more of a network element, a network service, a slice subnet, a slice manager, a cloud management as a service (CMAAS) manager, a bare metal unit, a bare metal as a service (BMAAS) manager, a service cluster, a policy life cycle manager, a hardware switch, a field programmable gate array, an eNodeB (eNB) in 4G, an NR base station (gNB) in 5G, a virtual distributed unit (vDU) and a virtual centralized unit (vCU), which are components of a radio access network (RAN) in 4G, a DU and a CU, which are components of the RAN in 5G, a virtual RAN (VRAN), a GC, an NFP, a logic cluster, a network node, a network device, an antenna, a communication layer, or other suitable network element. In some embodiments, the one or more workflows corresponds to a type of network element and comprises one or more of an operating system installation, an upgrade, a firmware upgrade, a change in BIOS settings, or some other suitable operation or event.

In some embodiments, the status of the network function is an operation state of a node in a communication network, and the operation comprises at least one of an alarm indicating the network node is inoperative and at least a portion of the communication network is down, or an instruction to heal the communication network by fixing or circumventing the network node that is inoperative.

In some embodiments, the correlation engine **111** determines the network is down or a network node is inoperable based on the event data and the lifecycle manager **119** is configured to output an instruction to heal the network such that the network function **129** is recognized by the observability framework **101** and the data received by the observability framework **101** is processed by the observability framework **101** and generates event data that is determined by the correlation engine **111** to satisfy the one or more conditions in the correlation policy. In some embodiments,

12

based on one or more rules included in the correlation policy stored in the second datastore **127**, the correlation engine **111** causes one or more event data values to be stored in the second datastore **127** for future use when determining if one or more unordered or ordered subsequent events occurs or does not occur in accordance with the conditions included in the correlation policy.

As a non-limiting example to aid in understanding the capabilities of system **100**, an implementation of a correlation policy associated with restarting a pod is provided. In this example, the correlation policy includes rules directed to relocating a pod and restarting a network function. The rule directed to relocating a pod includes conditions for a quantity of times the pod has been restarted, a hold time, an action code to be implemented for relocating the pod, a topology root type such as “network function,” filters such as a particular location or location range, and a defined action mode for determining an order of events or for implementing defined actions, for example, “sequence.” The rule directed to restarting the network function includes conditions for a quantity of times the pod has been relocated, a hold time, an action code such as “restart network function,” a topology root type such as “network function,” filters such as a particular location or location range, and an action mode such as “topology.”

If a kubernetes cluster associated with system **100**, in this example, detects a failure of a container in a pod based on a probe signal sent to the pod by a kubernetes node in communication with the kubernetes cluster, the kubernetes node communicates the detected failure to the observability framework **101**. The observability framework **101** then detects pod failures within a preset time interval. The kubernetes cluster causes the container in the pod to be restarted and determines whether the restart was successful. The kubernetes cluster then communicates the event data indicative of whether the pod container or the pod was restarted. The kubernetes node sends the probe signal to the pod to determine whether the container in the pod is active or in failure, and the kubernetes cluster reports to the observability framework **101** if the kubernetes cluster detects a subsequent failure of the container in the pod.

If the observability framework **101** detects that the quantity of pod container or pod restarts within the preset time interval exceeds a threshold value, then the lifecycle manager **119** causes the pod to be relocated to a different node by, for example, outputting a relocate pod instruction to be implemented by an end-to-end orchestrator in communication with the lifecycle manager **119** or within which the lifecycle manager **119** is included. The end-to-end orchestrator then outputs an instruction to the kubernetes cluster to restart the pod container or restart the pod, and the kubernetes cluster then restarts the pod container or restarts the pod on a different node. In some embodiments, the correlation policy additionally includes a rule that causes the network function to be restarted based on a determination that relocating the pod container or the pod to a different node did not resolve the detected pod container or pod failure.

As another non-limiting example to aid in understanding the capabilities of system **100**, an implementation of a correlation policy associated with failing to heal a failed pod is provided. In this example, the correlation policy includes rules directed to detecting whether a pod faulted multiple times and escalating a reported failure to an administrator. The rule directed to detecting whether a pod faulted multiple times includes conditions for a quantity of times the pod faulted, a hold time, an action code to be implemented for restarting a network function, a topology root type such as

13

“network function,” filters such as a particular location or location range, and a defined action mode such as “topology.” The rule directed to escalating a reported failure to an administrator includes conditions for a quantity of times the pod has faulted, a hold time, an action code such as “notify administrator,” a topology root type such as “network function,” filters such as a particular location or location range, and an action mode such as “topology.”

If a kubernetes cluster associated with system 100, in this example, detects a failure of a container in a pod based on a probe signal sent to the pod by a kubernetes node in communication with the kubernetes cluster, the kubernetes cluster attempts to restart the failed pod container or pod a preset quantity of times. In some embodiments, the kubernetes cluster attempts to restart the failed pod container or pod a preset quantity of times and if unsuccessful, attempts to relocate the failed pod container or pod. In some embodiments, the kubernetes cluster concurrently attempts to restart the failed pod container or pod and attempts to relocate the failed pod container or pod. In some embodiments, the preset quantity of attempts to restart the failed pod container or the failed pod includes the attempts to relocate the failed pod container or the failed pod. In some embodiments, the kubernetes cluster attempts to separately relocate the failed pod container or the failed pod for a preset correspond quantity of times associated with relocating the failed pod container or the failed pod. If the kubernetes is unsuccessful at restarting and/or relocating the failed pod container or the failed pod within the preset quantity of attempts, the kubernetes cluster communicates event data to the observability framework 101 indicating the failure to restart and/or relocate the failed pod container or the failed pod.

The lifecycle manager 119, based on processing of the event data received by the observability framework 101, recognizes that the pod container or pod that failed to restart has resulted in a network function that is missing the failed pod container or failed pod. The lifecycle manager 119 then terminates the network function and restarts the network function by outputting a restart network function instruction to be implemented by an end-to-end orchestrator in communication with the lifecycle manager 119 or within which the lifecycle manager 119 is included. The end-to-end orchestrator then outputs an instruction to the kubernetes cluster to restart the network function. If the kubernetes cluster is unable to restart the network function, or the kubernetes cluster again detects that the pod container or the pod failed a preset quantity of times within a predefined period after restarting the network function, the kubernetes cluster communicates event data to the observability framework 101 indicating that restarting the network function did not resolve the failed pod container or the failed pod, and the lifecycle manager 119 is caused to output an alert message to a system administrator as an alarm.

In some embodiments, the one or more rules included in a correlation policy involve one or more ordered and/or unordered sequences of events that are processed linearly or in parallel to determine the functionality of any defined network function included in a rule or combination of rules customized by a user by way of user interface 125. In some embodiments, one or more rules are processed independently or in conjunction with one another within a preset period of time.

EXAMPLES

The following non-limiting Examples 1-6 are provided to demonstrate NFAT evaluation wherein correlation engine

14

111 processes event data which, for simplicity, is discussed with respect to basic example variables “A” and “B” for exemplary correlation policies which, as discussed, may be customized for any particular purpose for identifying whether an operation is to be caused to occur or not. The event data in the following examples is merely exemplary for discussion purposes and can be any combination of variables, enriched or not enriched, associated with event data.

In Examples 1-6, the event data includes an event stream that comprises events A_1A_2B . Correlation engine 111 processes the event data according to the correlation policies stored in second datastore 127.

Example 1

In this example, the correlation policy is SC ($A>0$, $B>0$).

The correlation engine 111 scans the event stream A_1A_2B from the first event, A_1 .

Since A_1 fulfills the order of “A first and B later” (i.e., the order defined by the SC policy), correlation engine 111 creates a first array, Array 1. In this example, since the SC policy defines two conditions, Array 1 will contain two columns, each of which associates with event A and event B, respectively. Generally, the initial state of the two columns are both in a NULL state when no associated events have been detected. In this case, since an event A is detected, the number of A in the first column of Array 1 will be increased to “1”, and the first column of Array 1 will have a TRUE state (since $A>0=TRUE$).

Array 1	
A = 1 $A > 0 = TRUE$	B = 0 $B > 0 = NULL$

Nevertheless, since the correlation engine 111 has not yet detected any event B, the number of B in the second column of Array 1 is “0” and the NULL state of second column of Array 1 is maintained (since $B>0=NULL$). Thus, although the condition “ $A>0$ ” in the first column of Array 1 is matched with one of the conditions defined by the SC policy (i.e., $A>0=TRUE$, the conditions defined by the SC policy are partially matched), in view that a hold time has not yet passed and $B>0=NULL$, the current state of Array 1 is considered by the correlation engine 111 as a NULL state (which is a result of the TRUE state of the first column and the NULL state of the second column). Accordingly, the correlation engine 111 will not cause an operation to occur and will continue scanning/evaluating the subsequent event in the event stream.

If an event B is detected by the correlation engine 111 within a predefined hold time (if there is one), then the second column of Array 1 will have a TRUE state (since $B>0=TRUE$ in that case), which will result in the state of Array 1 being the TRUE state. Accordingly, the correlation engine 111 will cause an operation to occur and will delete Array 1 thereafter. On the other hand, if no event B is detected after the hold time, the correlation engine 111 will consider the NULL state of Array 1 as a FALSE state, and thus consider the final state of Array 1 as a FALSE state. Accordingly, the correlation engine 111 will delete Array 1 without causing any operation to occur.

Example 2

This example is a continuation of Example 1, wherein the correlation policy is SC ($A>0$, $B>0$), and the processing of

15

the event stream by the correlation engine 111 is within a predefined hold time. After completion of scanning of the first event of the event stream, correlation engine 111 will continue scanning the next event in the event stream. Thus, the correlation engine 111 scans the next event in the event stream A_1A_2B , A_2 .

Since A_2 fulfills the order of “A first and B later” (i.e., the order defined by the SC policy), and since the SC policy requires all events to be adjacent to each other (i.e., to be included into the same column in Array 1), the number of A in the first column of Array 1 will be increased to “2”.

Array 1	
A = 2 A > 0 = TRUE	B = 0 B > 0 = NULL

The correlation engine 111 will scan the next event in the event stream A_1A_2B , B.

Since B fulfills the order of “A first and B later” (i.e., the order defined by the SC policy), and since the second column of Array 1 is waiting for an event B, the number of B in the second column of Array 1 will be increased to “1”, and the second column of Array 1 will have a TRUE state (since $B>0=TRUE$).

Array 1	
A = 2 A > 0 = TRUE	B = 1 B > 0 = TRUE

Since both first and second columns are now in a TRUE state (which will result in the state of Array 1 being the TRUE state), all conditions defined by the SC policy are matched, Array 1 fully matches the conditions of SC ($A>0$, $B>0$), and correlation engine 111 causes an operation to occur regardless of whether or not it is still within the hold time. Subsequently, the correlation engine 111 deletes Array 1.

Example 3

In this example, the correlation policy is SNC ($A>0$, $B>0$).

The correlation engine 111 scans the event stream A_1A_2B from the first event, A_1 .

Since A_1 fulfills the order of “A first and B later” (i.e., the order defined by the SNC policy), the correlation engine 111 creates a first array, Array 1. In this example, since the SNC policy defines two conditions, Array 1 will contain two columns, each of which associates with event A and event B, respectively. Since an event A is detected, the number of A in the first column of Array 1 will be increased to “1”, and the first column of Array 1 will have a TRUE state (since $A>0=TRUE$).

Array 1	
A = 1 A > 0 = TRUE	B = 0 B > 0 = NULL

Since the correlation engine 111 has not yet detected any event B, the number of B in the second column of Array 1 is “0” and the NULL state of second column of Array 1 is maintained (since $B>0=NULL$). Thus, although the condi-

16

tion “ $A>0$ ” in the first column of Array 1 is matched with one of the conditions defined by the SNC policy (i.e., $A>0=TRUE$, the conditions defined by the SNC policy are partially matched), in view that a hold time has not yet passed and $B>0=NULL$, the correlation engine 111 considers the current state of Array 1 as a NULL state (which is a result of the TRUE state in the first column and the NULL state in the second column). Accordingly, the correlation engine 111 will not cause an operation to occur, and the correlation engine 111 will continue scanning/evaluating the event stream.

The correlation engine 111 then scans the next event in the event stream A_1A_2B , A_2 .

Since A_2 fulfills the order of “A first and B later” (i.e., the order defined by the SNC policy), and since the SNC policy does not require all events to be adjacent to each other and the condition “ $A>0$ ” in the first column of Array 1 is already matched (i.e., the conditions defined by the SNC policy are partially matched by Array 1), the correlation engine 111 creates a second array, Array 2. Similar to Array 1, since the SNC policy defines two conditions, Array 2 also contains two columns, each of which associates with event A and event B, respectively. Since an event A is detected, the number of A in the first column of Array 2 will be increased to “1”, and the first column of Array 2 will have a TRUE state (since $A>0=TRUE$).

Array 1	
A = 1 A > 0 = TRUE	B = 0 B > 0 = NULL

Array 2	
A = 1 A > 0 = TRUE	B = 0 B > 0 = NULL

Since the correlation engine 111 has not yet detected any event B, the number of B in the second column of Array 1 and Array 2 is “0” and the NULL state of second column of Array 1 and Array 2 is maintained (since $B>0=NULL$). Thus, although the condition “ $A>0$ ” in column 1 of Array 1 and Array 2 is matched with one of the conditions defined by the SNC policy (i.e., $A>0=TRUE$, the conditions defined by the SNC policy are partially matched), in view that the hold time has not yet passed and $B>0=NULL$ in both Array 1 and Array 2, the current state of Array 1 and Array 2 is considered by the correlation engine 111 as a NULL state (which is a result of the TRUE state and the NULL state). Accordingly, the correlation engine 111 will not cause an operation to occur and will continue scanning/evaluating the event stream.

The correlation engine 111 will then scan the next event in the event stream A_1A_2B , B.

Since B fulfills the order of “A first and B later” (i.e., the order defined by the SNC policy), and since the second column of Array 1 is waiting for an event B and Array 1 is created before Array 2, the number of B in the second column of Array 1 will be increased to “1”, and the second column of Array 1 will have a TRUE state (since $B>0=TRUE$). The state of Array 2 remains unchanged.

17

Array 1	
A = 1 A > 0 = TRUE	B = 1 B > 0 = TRUE
Array 2	
A = 1 A > 0 = TRUE	B = 0 B > 0 = NULL

Since both of the first and second columns in Array 1 are in a TRUE state (which will result in the state of Array 1 being the TRUE state), all conditions defined by the SC policy is matched, Array 1 fully matches the conditions of SNC (A>0, B>0), the correlation engine 111 causes an operation to occur regardless of whether or not it is still within the hold time. Subsequently, the correlation engine deletes Array 1, and only Array 2 remains.

Array 2	
A = 1 A > 0 = TRUE	B = 0 B > 0 = NULL

Since the conditions of Array 2 only partially match the conditions defined by the SNC policy, the correlation engine 111 will maintain Array 2 for the hold time and will wait for the next event.

If an event B is detected by the correlation engine 111 within the hold time, then the second column of Array 2 will have a TRUE state (since B>0=TRUE in that case), which will result in the state of Array 2 being the TRUE state. Accordingly, the correlation engine 111 causes an operation to occur and deletes Array 2 thereafter. On the other hand, if no event B is detected after the hold time, the correlation engine 111 considers the NULL state of Array 2 as a FALSE state, and thus considers the final state of Array 2 as a FALSE state. Accordingly, the correlation engine 111 deletes Array 2 without causing an operation to occur.

Example 4

In this example, the correlation policy is RC (A>0 AND B>0).

The correlation engine 111 scans the event stream A₁A₂B from the first event, A₁.

Since the conditions are defined by the RC policy, and the first event A₁ is associated with one of the conditions (i.e., A>0), the correlation engine 111 creates a tree structure, Tree 1. In this example, since the RC policy defines two conditions (i.e., A>0 and B>0), two leaves will be created for Tree 1.

Generally, the state of the two leaves are both in a NULL state when no associated events have been detected. In this case, since an event A is detected, the number of A in Leaf 1 is increased into "1", and the Leaf 1 will have a TRUE state (since A>0=TRUE).

Tree 1	
Leaf 1	Leaf 2
A = 1 A > 0 = TRUE	B = 0 B > 0 = NULL

18

Generally, before passing a hold time, the final state of Tree 1 has not yet been evaluated by the correlation engine 111, and thus the state of Tree 1 can be either one of the states of TRUE, FALSE, or NULL. After the hold time, the correlation engine 111 evaluates the final state of Tree 1, and the final state will only be one of the TRUE state or the FALSE state.

In this example, the correlation engine 111 has not yet detected any event B, and thus the number of B in Leaf 2 is "0" and the condition of "B>0" in Leaf 2 is having a NULL state. Although the condition "A>0" in Leaf 1 is matched with one of the conditions defined by the RC policy (i.e., A>0=TRUE, the conditions defined by the RC policy are partially matched), in view that the hold time has not yet passed and B>0=NULL in Leaf 2, the current state of Tree 1 will be considered by the correlation engine 111 as a NULL state (which is a result of the TRUE state in Leaf 1 and the NULL state in Leaf 2). Accordingly, the correlation engine 111 will not cause an operation to occur and will continue scanning/evaluating the event stream.

Subsequently, if an event B is detected by the correlation engine 111 within the hold time, then the state of Leaf 2 will become a TRUE state (since B>0=TRUE), which will result in the state of Tree 1 being the TRUE state. Accordingly, the correlation engine 111 will cause an operation to occur and will delete Tree 1 thereafter. On the other hand, if no event B is detected after the hold time, the correlation engine 111 considers the NULL state of Tree 1 as a FALSE state, and thus considers the final state of Tree 1 as a FALSE state. Accordingly, the correlation engine 111 deletes Tree 1 without causing an operation to occur.

Example 5

This example is a continuation of Example 4, wherein the correlation policy is RC (A>0 AND B>0), and the processing of the event stream by the correlation engine 111 is within the hold time. After completion of scanning of the first event of the event stream, correlation engine 111 will continue scanning the next event in the event stream. Thus, the correlation engine 111 scans the next event in the event stream A₁A₂B, A₂.

Since RC policy requires that all events be adjacent to each other (i.e., in the same leaf of the same tree structure), the number of A in Leaf 1 will be increased to "2".

Tree 1	
Leaf 1	Leaf 2
A = 2 A > 0 = TRUE	B = 0 B > 0 = NULL

The correlation engine 111 then scans the next event in the event stream A₁A₂B, B.

Since Leaf 2 is waiting for event B, the number of B in Leaf 2 will be increased to "1", and Leaf 2 will have a TRUE state (since B>0=TRUE).

Tree 1	
Leaf 1	Leaf 2
A = 2 A > 0 = TRUE	B = 1 B > 0 = TRUE

19

Since both of Leaf 1 and Leaf 2 are in the TRUE state (which will result in the state of Tree 1 being the TRUE state), all conditions defined by the RC policy is matched, the state/conditions of Tree 1 fully match the conditions of RC ($A > 0$ AND $B > 0$), and the correlation engine **111** causes an operation to occur. Subsequently, the correlation engine **111** deletes Tree 1.

Example 6

In this example, the correlation policy is RNC ($A > 0$ AND $B > 0$).

Correlation engine **111** scans the event stream A_1A_2B from the first event, A_1 .

Since the conditions are defined by RNC policy, and the first event A_1 is associated with one of the conditions (i.e., $A > 0$), the correlation engine **111** creates a first tree structure, Tree 1. In this example, since the RC policy defines two conditions (i.e., $A > 0$ and $B > 0$), the correlation engine **111** creates two leaves for the first tree.

The number of A in Leaf 1 of Tree 1 is increased into “1”, and Leaf 1 of Tree 1 will have a TRUE state (since $A > 0 = \text{TRUE}$).

Tree 1	
Leaf 1	Leaf 2
A = 1 $A > 0 = \text{TRUE}$	B = 0 $B > 0 = \text{NULL}$

Since the correlation engine has not yet detected any event B, the number of B in Leaf 2 is “0” and the condition of “ $B > 0$ ” is having a NULL state. Thus, although the condition “ $A > 0$ ” in Leaf 1 is matched with one of the conditions defined by the RNC policy (i.e., $A > 0 = \text{TRUE}$, the conditions defined by the RNC policy are partially matched), and in view that the hold time has not yet passed and $B > 0 = \text{NULL}$ in Leaf 2 of Tree 1, the correlation engine **111** considers the current state of Tree 1 as a NULL state (which is a result of the TRUE state and the NULL state). Accordingly, the correlation engine **111** will not cause an operation to occur and will continue scanning/evaluating the event stream.

The correlation engine **111** then scans the next event in the event stream A_1A_2B , A_2 .

Since the RNC policy does not require all events to be adjacent to each other (i.e., in the same Leaf of the same Tree structure), and since A_2 is associated with one of the conditions (i.e., $A > 0$) defined by the RNC policy and the condition “ $A > 0$ ” in Leaf 1 of Tree 1 is already matched (i.e., the conditions defined by the RNC policy are partially matched by Tree 1), the correlation engine **111** creates a second tree structure, Tree 2. In this example, since the RC policy defines two conditions (i.e., $A > 0$ and $B > 0$), the correlation engine **111** creates two leaves for the second tree, Tree 2.

The number of A in Leaf 1 of Tree 2 is increased into “1”, and Leaf 1 of Tree 2 will have a TRUE state (since $A > 0 = \text{TRUE}$).

Tree 1	
Leaf 1	Leaf 2
A = 1 $A > 0 = \text{TRUE}$	B = 0 $B > 0 = \text{NULL}$

20

Tree 2	
Leaf 1	Leaf 2
A = 1 $A > 0 = \text{TRUE}$	B = 0 $B > 0 = \text{NULL}$

Since the correlation engine **111** has not yet detected any event B, the number of B in Leaf 2 of Tree 1 and Tree 2 is “0” and the condition of “ $B > 0$ ” in Leaf 2 of Tree 1 and Tree 2 is having a NULL state. Thus, although the condition “ $A > 0$ ” in Leaf 1 of Tree 1 and Leaf 1 of Tree 2 is matched with one of the conditions defined by the RNC policy (i.e., $A > 0 = \text{TRUE}$, the conditions defined by the RC policy are partially matched), and in view that the hold time has not yet passed and $B > 0 = \text{NULL}$ in Leaf 2 of Tree 1 and Leaf 2 of Tree 2, the correlation engine **111** considers the current state of Tree 1 and Tree 2 as a NULL state (which is a result of the TRUE state and the NULL state). Accordingly, the correlation engine **111** will not cause an operation to occur and will continue scanning/evaluating the event stream.

The correlation engine **111** then scans the next event in the event stream A_1A_2B , B.

Since Leaf 2 of Tree 1 is waiting for an event B and since Tree 1 is created earlier than Tree 2, the number of B in Leaf 2 of Tree 1 will increase to “1”, and Leaf 2 of Tree 1 will have a TRUE state (since $B > 0 = \text{TRUE}$). The status of Tree 2 will remain unchanged.

Tree 1	
Leaf 1	Leaf 2
A = 1 $A > 0 = \text{TRUE}$	B = 1 $B > 0 = \text{TRUE}$

Tree 2	
Leaf 1	Leaf 2
A = 1 $A > 0 = \text{TRUE}$	B = 0 $B > 0 = \text{NULL}$

Since both of Leaf 1 and Leaf 2 of Tree 1 is in the TRUE state (which will result in Tree 1 being in the TRUE state), all conditions defined by the RNC policy are matched, and conditions of Tree 1 fully match the conditions of RNC ($A > 0$ AND $B > 0$), the correlation engine **111** causes an operation to occur. Subsequently, the correlation engine **111** deletes Tree 1.

Since the conditions of Tree 2 only partially match the conditions defined by the RNC policy, the correlation engine **111** will maintain Tree 2 for the hold time and wait for the next event.

If an event B is detected by the correlation engine **111** within the hold time, then Leaf 2 of Tree 2 will have a TRUE state (since $B > 0 = \text{TRUE}$ in that case), which will result in the state of Tree 2 being the TRUE state. Accordingly, the correlation engine **111** causes an operation to occur and deletes Tree 2 thereafter. On the other hand, if no event B is detected after the hold time, the correlation engine will consider the NULL state of Tree 2 as a FALSE state, and thus consider the final state of Tree 2 as a FALSE state. Accordingly, the correlation engine deletes Tree 2 without causing an operation to occur.

21

The following non-limiting Examples 7-11 are provided to demonstrate NFAT evaluation wherein correlation engine **111** processes event data which, for simplicity, is discussed with respect to basic example variables “A”, “B” and “C” for exemplary correlation policies which, as discussed, may be customized for any particular purpose for identifying whether an operation is to be caused to occur or not. The event data in the following examples is merely exemplary for discussion purposes and can be any combination of variables, enriched or not enriched, associated with event data.

In Examples 7-11, the event data includes an event stream that comprises events A_1A_2CB . Correlation engine **111** processes the event data according to the correlation policies stored in second datastore **127**.

Examples 7-11 involve different policy types, including a strict contiguous (SC) policy, a strict non-contiguous (SNC) policy, a relaxed contiguous (RC) policy, and a relaxed non-contiguous (RNC) policy.

Strict-type policies (e.g., SC and SNC), as discussed, require all events to occur in a specific order. Thus, inherently, evaluation which involve strict-type policies is performed based on AND-logic only. On the other hand, since relaxed-type policies do not require the events to occur in a specific order, evaluation which involve relaxed-type policies can be performed based on AND-logic, OR-logic, or a combination thereof. Thus, strict-type policies may or may not include a logic-gate, while relaxed-type policies include a logic-gate. For example, strict-type policies are presented in the form of SC (A, B) and SNC (A,B), which inherently means that SC (A, B) and SNC (A, B) should be evaluated based on AND-logic, while relaxed-type policies are presented in the form of RC (A [Logic-gate] B), and RNC (A [Logic-gate] B), e.g., RC (A AND B), RNC (A OR B), and the like.

Example 7

In this example, the correlation policy is SC ($A>0$, $B>0$), and the correlation engine **111** receives the event stream without event filtering or enrichment.

The correlation engine **111** scans the event stream A_1A_2CB from the first event, A_1 .

Since A_1 fulfills the order of “A first and B later” (i.e., order defined by the SC policy), Array 1 will be created. In this example, since the SC policy defines two conditions, Array 1 will contain two columns, each of which associates with event A and event B, respectively. Generally, the initial state of the two columns are both in a NULL state when no associated events have been detected. In this case, since an event A is detected, the number of A in the first column of Array 1 will be increased to “1”, and the first column of Array 1 will have a TRUE state (since $A>0=TRUE$).

Array 1	
A = 1	B = 0
$A > 0 = TRUE$	$B > 0 = NULL$

Nevertheless, since the correlation engine **111** has not yet detected any event B, the number of B in the second column of Array 1 is “0” and the NULL state of second column of Array 1 is maintained (since $B>0=NULL$). Thus, although the condition “ $A>0$ ” in the first column of Array 1 is matched with one of the conditions defined by the SC policy (i.e., $A>0=TRUE$, the conditions defined by the SC policy

22

are partially matched), in view that the hold time has not yet passed and $B>0=NULL$, the current state of Array 1 will be considered as a NULL state (which is a result of TRUE state of the first column and NULL state of the second column). Accordingly, the correlation engine **111** will not cause an operation to occur and will continue scanning/evaluating the subsequent event in the event stream.

If an event B is detected by the correlation engine **111** within a predefined hold time (if there is one), then the second column of Array 1 will have a TRUE state (since $B>0=TRUE$ in that case), which will result in the state of Array 1 being the TRUE state. Accordingly, the correlation engine **111** will cause an operation to occur and will delete Array 1 thereafter. On the other hand, if no event B is detected after the hold time, the correlation engine **111** will consider the NULL state of Array 1 as a FALSE state, and thus consider the final state of Array 1 as a FALSE state. Accordingly, the correlation engine **111** will delete Array 1 without causing any operation to occur. In the following examples, the correlation engine **111** is within the hold time, and the correlation engine **111** will continue scanning the next event in the event stream.

The correlation engine **111** will then scan the next event in the event stream A_1A_2CB , A_2 .

Since A_2 fulfills the order of “A first and B later” (i.e., the order defined by the SC policy), and since the SC policy requires that all events to be adjacent to each other (i.e., to be included into the same column in Array 1), the number of A in the first column of Array 1 will be increased to “2”.

Array 1	
A = 2	B = 0
$A > 0 = TRUE$	$B > 0 = NULL$

The correlation engine **111** will scan the next event in the event stream A_1A_2CB , C.

Since event C is not an event which is related to the conditions defined by the SC policy, and the contiguous nature of SC policy requires similar types of events to be adjacent to each other in one array, but Array 1 does not contain a column for event C, the correlation engine **111** will delete Array 1 without causing an operation to occur.

The correlation engine **111** will then scan the next event in the event stream A_1A_2CB , B.

Since event B is now considered as the first event (i.e., at this point of scanning the data stream with Array 1 being deleted), event B does not fulfill the order of “A first and B later” (i.e., the order defined by the SC policy). Accordingly, the correlation engine **111** will ignore event B and will not create any array.

Example 8

In this example, the correlation policy is SNC ($A>0$, $B>0$), and the correlation engine **111** receives the event stream without event filtering or enrichment.

The correlation engine **111** scans the event stream A_1A_2CB from the first event, A_1 .

Since A_1 fulfills the order of “A first and B later” (i.e., the order defined by the SNC policy), the correlation engine **111** creates a first array, Array 1. In this example, since the SNC policy defines two conditions, Array 1 will contain two columns, each of which associates with event A and event B, respectively. Since an event A is detected, the number of A

23

in the first column of Array 1 will be increased to “1”, and the first column of Array 1 will have a TRUE state (since $A > 0 = \text{TRUE}$).

Array 1	
A = 1 $A > 0 = \text{TRUE}$	B = 0 $B > 0 = \text{NULL}$

Since the correlation engine **111** has not yet detected any event B, the number of B in the second column of Array 1 is “0” and the NULL state of second column of Array 1 is maintained (since $B > 0 = \text{NULL}$). Thus, although the condition “ $A > 0$ ” in the first column of Array 1 is matched with one of the conditions defined by the SNC policy (i.e., $A > 0 = \text{TRUE}$, the conditions defined by the SNC policy are partially matched), in view that a hold time has not yet passed and $B > 0 = \text{NULL}$, the correlation engine **111** considers the current state of Array 1 as a NULL state (which is a result of the TRUE state in the first column and the NULL state in the second column). Accordingly, the correlation engine **111** will not cause an operation to occur, and the correlation engine **111** will continue scanning/evaluating the event stream.

The correlation engine **111** then scans the next event in the event stream $A_1 A_2 C B, A_2$.

Since A_2 fulfills the order of “A first and B later” (i.e., the order defined by the SNC policy), and since the SNC policy does not require all events to be adjacent to each other and the condition “ $A > 0$ ” in the first column of Array 1 is already matched (i.e., the conditions defined by the SNC policy are partially matched by Array 1), the correlation engine **111** creates a second array, Array 2. Similar to Array 1, since the SNC policy defines two conditions, Array 2 also contains two columns, each of which associates with event A and event B, respectively. Since an event A is detected, the number of A in the first column of Array 2 will be increased to “1”, and the first column of Array 2 will have a TRUE state (since $A > 0 = \text{TRUE}$).

Array 1	
A = 1 $A > 0 = \text{TRUE}$	B = 0 $B > 0 = \text{NULL}$

Array 2	
A = 1 $A > 0 = \text{TRUE}$	B = 0 $B > 0 = \text{NULL}$

Since the correlation engine **111** has not yet detected any event B, the number of B in the second column of Array 1 and Array 2 is “0” and the NULL state of second column of Array 1 and Array 2 is maintained (since $B > 0 = \text{NULL}$). Thus, although the condition “ $A > 0$ ” in column 1 of Array 1 and Array 2 is matched with one of the conditions defined by the SNC policy (i.e., $A > 0 = \text{TRUE}$, the conditions defined by the SNC policy are partially matched), in view that the hold time has not yet passed and $B > 0 = \text{NULL}$ in both Array 1 and Array 2, the current state of Array 1 and Array 2 is considered by the correlation engine **111** as a NULL state (which is a result of the TRUE state and the NULL state).

24

Accordingly, the correlation engine **111** will not cause an operation to occur and will continue scanning/evaluating the event stream.

The correlation engine **111** will then scan the next event in the event stream $A_1 A_2 C B, C$.

Event C is not an event which is related to the conditions defined by the SNC policy, and none of Array 1 and Array 2 contain a column related to event C. The difference between SNC policy based evaluation and SC policy based evaluation is that, in SNC policy based evaluation, instead of deleting the array as in Example 7 of the SC policy based evaluation, the correlation engine **111** will ignore event C and continue the process in this example.

The correlation engine **111** will then scan the next event in the event stream $A_1 A_2 C B, B$.

Since B fulfills the order of “A first and B later” (i.e., the order defined by the SNC policy), and since the second column of Array 1 is waiting for an event B and Array 1 is created before Array 2, the number of B in the second column of Array 1 will be increased to “1”, and the second column of Array 1 will have a TRUE state (since $B > 0 = \text{TRUE}$). The state of Array 2 remains unchanged.

Array 1	
A = 1 $A > 0 = \text{TRUE}$	B = 1 $B > 0 = \text{TRUE}$

Array 2	
A = 1 $A > 0 = \text{TRUE}$	B = 0 $B > 0 = \text{NULL}$

Since both of the first and second columns in Array 1 are in a TRUE state (which will result in the state of Array 1 being the TRUE state), all conditions defined by the SC policy is matched, Array 1 fully matches the conditions of SNC ($A > 0, B > 0$), the correlation engine **111** causes an operation to occur regardless of whether or not it is still within the hold time. Subsequently, the correlation engine **111** deletes Array 1, and only Array 2 remains.

Array 2	
A = 1 $A > 0 = \text{TRUE}$	B = 0 $B > 0 = \text{NULL}$

Since the conditions of Array 2 only partially match the conditions defined by the SNC policy, the correlation engine **111** will maintain Array 2 for the hold time and will wait for the next event.

If an event B is detected by the correlation engine **111** within the hold time, then the second column of Array 2 will have a TRUE state (since $B > 0 = \text{TRUE}$ in that case), which will result in the state of Array 2 being the TRUE state. Accordingly, the correlation engine **111** causes an operation to occur and deletes Array 2 thereafter. On the other hand, if no event B is detected after the hold time, the correlation engine **111** considers the NULL state of Array 2 as a FALSE state, and thus considers the final state of Array 2 as a FALSE state. Accordingly, the correlation engine **111** deletes Array 2 without causing an operation to occur.

25

Example 9

In this example, the correlation policy is RC ($A > 0$ AND $B > 0$), and the correlation engine 111 receives the event stream without event filtering or enrichment.

The correlation engine 111 scans the event stream A_1A_2CB from the first event, A_1 .

Since the conditions are defined by the RC policy, and the first event A_1 is associated with one of the conditions (i.e., $A > 0$), the correlation engine 111 creates a first tree structure, Tree 1. In this example, since the RC policy defines two conditions (i.e., $A > 0$ and $B > 0$), two leaves will be created for Tree 1.

Since the conditions are defined by the RC policy, and the first event A_1 is associated with one of the conditions (i.e., $A > 0$), the correlation engine 111 creates a tree structure, Tree 1. In this example, since the RC policy defines two conditions (i.e., $A > 0$ and $B > 0$), two leaves will be created for Tree 1.

Generally, the state of the two leaves are both in a NULL state when no associated events have been detected. In this case, since an event A is detected, the number of A in Leaf 1 is increased into "1", and the Leaf 1 will have a TRUE state (since $A > 0 = \text{TRUE}$).

Tree 1	
Leaf 1	Leaf 2
A = 1 A > 0 = TRUE	B = 0 B > 0 = NULL

Generally, before passing a hold time, the final state of Tree 1 has not yet been evaluated by the correlation engine 111, and thus the state of Tree 1 can be either one of the states of TRUE, FALSE, or NULL. After the hold time, the correlation engine 111 evaluates the final state of Tree 1, and the final state will only be one of the TRUE state or the FALSE state.

In this example, since this evaluation is performed on an AND-logic basis and the correlation engine 111 has not yet detected any event B, and thus the number of B in Leaf 2 is "0" and the condition of " $B > 0$ " in Leaf 2 is having a NULL state. Although the condition " $A > 0$ " in Leaf 1 is matched with one of the conditions defined by the RC policy (i.e., $A > 0 = \text{TRUE}$, the conditions defined by the RC policy are partially matched), in view that the hold time has not yet passed and $B > 0 = \text{NULL}$ in Leaf 2, the current state of Tree 1 will be considered as a NULL state (which is a result of TRUE state in Leaf 1 and NULL state in Leaf 2). Accordingly, the correlation engine 111 will not cause an operation to occur and will continue scanning/evaluating the event stream.

Subsequently, if an event B is detected by the correlation engine 111 within the hold time, then the state of Leaf 2 will become a TRUE state (since $B > 0 = \text{TRUE}$), which will result in the state of Tree 1 being a TRUE state. Accordingly, the correlation engine 111 will cause an operation to occur and will delete Tree 1 thereafter. On the other hand, if no event B is detected after the hold time, the correlation engine 111 will consider the NULL state of Tree 1 as a FALSE state, and thus consider the final state of Tree 1 as a FALSE state. Accordingly, the correlation engine 111 deletes Tree 1 without causing an operation to occur.

Example 10

This example is a continuation of Example 9, wherein the correlation policy is RC ($A > 0$ AND $B > 0$), and the process-

26

ing of the event stream by the correlation engine 111 is within the hold time. After completion of scanning of the first event of the event stream, correlation engine 111 will continue scanning the next event in the event stream. Thus, the correlation engine 111 scans the next event in the event stream A_1A_2CB , A_2 .

Since RC policy requires that all events to be adjacent to each other (i.e., in the same leaf of the same tree structure), the number of A in Leaf 1 will be increased to "2".

Tree 1	
Leaf 1	Leaf 2
A = 2 A > 0 = TRUE	B = 0 B > 0 = NULL

The correlation engine 111 then scans the next event in the event stream A_1A_2CB , C.

Since event C is not an event which is related to the conditions defined by the RC policy, and the contiguous nature of RC policy requires similar types of events to be adjacent to each other in one tree structure but Tree 1 does not contain a column for event C, the correlation engine 111 will delete Tree 1 without causing an operation to occur.

The correlation engine 111 then scans the next event in the event stream A_1A_2CB , B.

Since event B is now considered as the first event (i.e., since Tree 1 is deleted), and event B is associated with one of the conditions (i.e., $B > 0$), a new tree structure Tree 2 will be created by the correlation engine 111. In this example, since the RC policy defines two conditions (i.e., $A > 0$ and $B > 0$), two leaves will be created for the tree.

The number of B in Leaf 2 will be increased to "1", and Leaf 2 will have a TRUE state (since $B > 0 = \text{TRUE}$).

Tree 2	
Leaf 1	Leaf 2
A = 0 A > 0 = NULL	B = 1 B > 0 = TRUE

Since this evaluation is performed on AND-logic basis and the correlation engine 111 has not yet detected any event A for Leaf 1 of Tree 2, and the number of A in Leaf 1 is "0" and the condition of " $A > 0$ " in Leaf 1 is having a NULL state. Although the condition " $B > 0$ " in Leaf 2 is matched with one of the conditions defined by the RC policy (i.e., $B > 0 = \text{TRUE}$, the conditions defined by the RC policy are partially matched), in view that the hold time has not yet passed and $A > 0 = \text{NULL}$ in Leaf 1, the current state of Tree 2 will be considered as a NULL state (which is a result of NULL state in Leaf 1 and TRUE state in Leaf 2). Accordingly, the correlation engine 111 will not cause an operation to occur and will continue scanning/evaluating the event stream until the hold time is passed or until all conditions are fully matched.

Example 11

In this example, the correlation policy is RNC ($A > 0$ AND $B > 0$), and the correlation engine 111 receives the event stream without event filtering or enrichment.

Correlation engine 111 scans the event stream A_1A_2CB , from the first event, A_1 .

27

Since the conditions are defined by RNC policy, and the first event A_1 is associated with one of the conditions (i.e., $A>0$), the correlation engine **111** creates a first tree structure, Tree 1. In this example, since the RNC policy defines two conditions (i.e., $A>0$ and $B>0$), the correlation engine **111** creates two leaves for the first tree structure.

The number of A in Leaf 1 of Tree 1 is increased into “1”, and Leaf 1 of Tree 1 will have a TRUE state (since $A>0=TRUE$).

Tree 1	
Leaf 1	Leaf 2
A = 1 A > 0 = TRUE	B = 0 B > 0 = NULL

Since the correlation engine **111** has not yet detected any event B, and thus the number of B in Leaf 2 is “0” and the condition of “ $B>0$ ” is having a NULL state. Further, since this evaluation is based on AND-logic, although the condition “ $A>0$ ” in Leaf 1 is matched with one of the conditions defined by the RNC policy (i.e., $A>0=TRUE$, the conditions defined by the RNC policy are partially matched), in view that the hold time has not yet passed and $B>0=NULL$ in Leaf 2 of Tree 1, the current state of Tree 1 will be considered as a NULL state (which is a result of TRUE state and NULL state). Accordingly, the correlation engine **111** will not cause any operation to occur and will continue scanning/evaluating the event stream.

The correlation engine **111** then scans the next event in the event stream A_1A_2CB, A_2 .

Since the RNC policy does not require all events to be adjacent to each other (i.e., in the same Leaf of the same Tree structure), and since A_2 is associated with one of the conditions (i.e., $A>0$), a second tree structure, Tree 2 will be created by the correlation engine **111**. In this example, since the RNC policy defines two conditions (i.e., $A>0$ and $B>0$), two leaves will be created for the second tree structure.

The number of A in Leaf 1 of Tree 2 is increased into “1”, and Leaf 1 of Tree 2 will have a TRUE state (since $A>0=TRUE$).

Tree 1	
Leaf 1	Leaf 2
A = 1 A > 0 = TRUE	B = 0 B > 0 = NULL

Tree 2	
Leaf 1	Leaf 2
A = 1 A > 0 = TRUE	B = 0 B > 0 = NULL

Since the correlation engine **111** has not yet detected any event B, and thus the number of B in Leaf 2 of Tree 1 and Tree 2 is “0” and the condition of “ $B>0$ ” in Leaf 2 of Tree 1 and Tree 2 is having a NULL state. Further, since this evaluation is based on AND-logic, although the condition “ $A>0$ ” in Leaf 1 of Tree 1 and Leaf 1 of Tree 2 is matched with one of the conditions defined by the RNC policy (i.e., $A>0=TRUE$, the conditions defined by the RNC policy are

28

partially matched), in view that the hold time has not yet passed and $B>0=NULL$ in Leaf 2 of Tree 1 and Leaf 2 of Tree 2, the current state of Tree 1 and Tree 2 will be considered as a NULL state (which is a result of the TRUE state and the NULL state). Accordingly, the correlation engine **111** will not cause an operation to occur and will continue scanning/evaluating the event stream.

The correlation engine **111** then scans the next event in the event stream A_1A_2CB, C .

Event C is not an event which is related to the conditions defined by the RNC policy, and none of Tree 1 and Tree 2 contain a leaf related to event C. The difference between the RNC policy and the RC policy is that, in the RNC policy, the events are not required to be adjacent to each other. Thus, instead of deleting Tree 1 as in Example 9, the correlation engine **111** will ignore event C and continue the process in this example.

The correlation engine **111** then scans the next event in the event stream A_1A_2CB, B .

Since Leaf 2 of Tree 1 is waiting for an event B and since Tree 1 is created earlier than Tree 2, the number of B in Leaf 2 of Tree 1 will increase to “1”, and Leaf 2 of Tree 1 will have a TRUE state (since $B>0=TRUE$). The status of Tree 2 will remain unchanged.

Tree 1	
Leaf 1	Leaf 2
A = 1 A > 0 = TRUE	B = 1 B > 0 = TRUE

Tree 2	
Leaf 1	Leaf 2
A = 1 A > 0 = TRUE	B = 0 B > 0 = NULL

Since both of Leaf 1 and Leaf 2 of Tree 1 is in the TRUE state (which will result in Tree 1 being in the TRUE state), all conditions defined by the RNC policy are matched, and conditions of Tree 1 fully match the conditions of RNC ($A>0$ AND $B>0$), the correlation engine **111** causes an operation to occur. Subsequently, the correlation engine **111** deletes Tree 1.

Since the conditions of Tree 2 only partially match the conditions defined by the RNC policy, the correlation engine **111** will maintain Tree 2 for the hold time and wait for the next event.

The following non-limiting Examples 12-15 are provided to demonstrate NFAT evaluation wherein correlation engine **111** processes event data which, for simplicity, is discussed with respect to basic example variables “A”, “B” and “C” for exemplary correlation policies which, as discussed, may be customized for any particular purpose for identifying whether an operation is to be caused to occur or not. The event data in the following examples is merely exemplary for discussion purposes and can be any combination of variables, enriched or not enriched, associated with event data.

In Examples 12-15, the event data includes an event stream that comprises events A_1CA_2B . Correlation engine **111** processes the event data according to the correlation policies stored in second datastore **127**.

Examples 12 and 13 are directed to OR-logic based NFAT evaluation without event filtering or enrichment. Examples 14 and 15 are directed to OR-logic based NFAT evaluation with event filtering (i.e., event C which is not related to the conditions defined by the policies will be filtered and will not be involved in the NFAT evaluations).

Example 12

In this example, the correlation policy is RC ($A > 1$ OR $B > 0$).

The correlation engine **111** scans the event stream A_1CA_2B from the first event, A_1 .

Since the conditions are defined by the RC policy, and the first event A_1 is associated with one of the conditions (i.e., $A > 1$), the correlation engine **111** creates a first tree structure, Tree 1. In this example, since the RC policy defines two conditions (i.e., $A > 1$ and $B > 0$), the correlation engine creates two leaves for the first tree.

The number of A in Leaf 1 of Tree 1 is increased into “1”, and Leaf 1 of Tree 1 will have a NULL state (since $A=1$ and not $A > 1$).

Tree 1	
Leaf 1	Leaf 2
A = 1 $A > 0 = \text{NULL}$	B = 0 $B > 0 = \text{NULL}$

The correlation engine **111** then scans the next event in the event stream A_1CA_2B , C.

Since event C is not an event which is related to the conditions defined by the RC policy, and the contiguous nature of RC policy requires similar types of events to be adjacent to each other in one tree structure, but Tree 1 does not contain a column for event C, the correlation engine **111** will delete Tree 1 without causing an operation to occur.

The correlation engine **111** then scans the next event in the event stream A_1CA_2B , A_2 .

Since A_2 is associated with one of the conditions (i.e., $A > 1$), a second tree structure, Tree 2, will be created by the correlation engine **111**. In this example, since the RC policy defines two conditions (i.e., $A > 1$ and $B > 0$), two leaves will be created for the second tree.

The number of A in Leaf 1 of Tree 2 is increased into “1”. The NULL state of Leaf 1 will be remained unchanged, and thus the current state of Tree 2 will be remained in NULL state.

Tree 2	
Leaf 1	Leaf 2
A = 1 $A > 1 = \text{NULL}$	B = 0 $B > 0 = \text{NULL}$

The correlation engine **111** then scans the next event in the event stream A_1CA_2B , B.

The number of B in Leaf 2 will be increased to “1”, and Leaf 2 will have a TRUE state (since $B > 0 = \text{TRUE}$).

Tree 2	
Leaf 1	Leaf 2
A = 1 $A > 1 = \text{NULL}$	B = 1 $B > 0 = \text{TRUE}$

Since this evaluation is performed on OR-logic basis and since the condition “ $B > 0$ ” in Leaf 2 is matched with one of the conditions defined by the RC policy (i.e., $B > 0 = \text{TRUE}$, the conditions defined by the RC policy are partially matched), Tree 2 will have a TRUE state and thus the correlation engine **111** causes an action to occur. Accordingly, the correlation engine **111** will then delete Tree 2.

Example 13

In this example, the correlation policy is RNC ($A > 1$ OR $B > 0$).

The correlation engine **111** scans the event stream A_1CA_2B from the first event, A_1 .

Since the conditions are defined by the RNC policy, and the first event A_1 is associated with one of the conditions (i.e., $A > 1$), the correlation engine **111** creates a first tree structure, Tree 1. In this example, since the RNC policy defines two conditions (i.e., $A > 1$ and $B > 0$), the correlation engine **111** creates two leaves for the first tree.

The number of A in Leaf 1 of Tree 1 is increased into “1”, and Leaf 1 of Tree 1 will have a NULL state (since $A=1$ is not $A > 1$).

Tree 1	
Leaf 1	Leaf 2
A = 1 $A > 1 = \text{NULL}$	B = 0 $B > 0 = \text{NULL}$

The correlation engine **111** then scans the next event in the event stream A_1CA_2B , C.

Since event C is not an event which is related to the conditions defined by the RNC policy, the correlation engine **111** will ignore event C and continue the process.

The correlation engine **111** then scans the next event in the event stream A_1CA_2B , A_2 .

The number of A in Leaf 1 of Tree 1 is increased into “2”, and Leaf 1 of Tree 1 will have a TRUE state (since $A > 1 = \text{TRUE}$).

Tree 1	
Leaf 1	Leaf 2
A = 2 $A > 1 = \text{TRUE}$	B = 0 $B > 0 = \text{NULL}$

Since this evaluation is performed on OR-logic basis and since the condition “ $A > 1$ ” in Leaf 1 is matched with one of the conditions defined by the RNC policy (i.e., $A > 1 = \text{TRUE}$, the conditions defined by the RNC policy are partially matched), Tree 1 will have a TRUE state and thus the correlation engine **111** will cause an action to occur. Accordingly, the correlation engine **111** will delete Tree 1.

After deleting Tree 1, the correlation engine **111** continues to scan the next event in the event stream A_1CA_2B , B.

31

Since event B is associated with one of the conditions (i.e., $B > 0$), the correlation engine **111** creates a second tree structure, Tree 2. In this example, since the RNC policy defines two conditions (i.e., $A > 1$ and $B > 0$), the correlation engine **111** creates two leaves for the second tree.

Since an event B is detected, the number of B in Leaf 2 is increased into "1".

Tree 1	
Leaf 1	Leaf 2
A = 0 A > 1 = NULL	B = 1 B > 0 = TRUE

Since "1" is greater than 0, the condition of " $B > 0$ " in Tree 2 will become TRUE.

Since this evaluation is performed on OR-logic basis and since the condition " $B > 0$ " in Leaf 2 is matched with one of the conditions defined by the RNC policy (i.e., $B > 0 = \text{TRUE}$, the conditions defined by the RNC policy are partially matched), Tree 2 will have a TRUE state and thus the correlation engine **111** will cause an operation to occur. Accordingly, the correlation engine **111** will delete Tree 2.

Example 14

In this example, the correlation policy is RC ($A > 1$ OR $B > 0$), and the correlation engine **111** receives the event stream, having been filtered to remove the C from the event stream A_1CA_2B with event filtering, and thus the event stream received by the correlation engine **111** is event stream A_1A_2B .

Correlation engine **111** scans the event stream A_1A_2B from the first event, A_1 .

Since the conditions are defined by the RC policy, and the first event A_1 is associated with one of the conditions (i.e., $A > 1$), the correlation engine **111** creates a first tree structure, Tree 1. In this example, since the RC policy defines two conditions (i.e., $A > 1$ and $B > 0$), the correlation engine **111** create two leaves.

Since an event A is detected, the number of A in Leaf 1 is increased into "1", and Leaf 1 of Tree 1 will have a NULL state (since $A=1$ is not $A > 1$).

Tree 1	
Leaf 1	Leaf 2
A = 1 A > 1 = NULL	B = 0 B > 0 = NULL

The correlation engine **111** then scans the next event in the event stream A_1A_2B , A_2 .

The number of A in Leaf 1 will be increased into "2". Accordingly, since "2" is greater than 1, the condition of " $A > 1$ " in Tree 1 will become TRUE.

Tree 1	
Leaf 1	Leaf 2
A = 2 A > 1 = TRUE	B = 0 B > 0 = NULL

32

Since this evaluation is performed on OR-logic basis and since the condition " $A > 1$ " in Leaf 1 is matched with one of the conditions defined by the RC policy (i.e., $A > 1 = \text{TRUE}$, the conditions defined by the RC policy are partially matched), Tree 1 will have a TRUE state and thus the correlation engine **111** will cause an operation to occur. Accordingly, the correlation engine **111** will delete Tree 1.

After deleting Tree 1, the correlation engine **111** continues to scan the next event in the event stream A_1A_2B , B.

Since event B is associated with one of the conditions (i.e., $B > 0$), the correlation engine **111** creates a second tree structure, Tree 2. In this example, since the RC policy defines two conditions (i.e., $A > 1$ and $B > 0$), the correlation engine **111** creates two leaves for the second tree.

Since an event B is detected, the number of B in Leaf 2 is increased into "1".

Tree 2	
Leaf 1	Leaf 2
A = 0 A > 0 = NULL	B = 1 B > 0 = TRUE

Since "1" is greater than 0, the condition of " $B > 0$ " in Tree 1 will become TRUE.

Since this evaluation is performed on OR-logic basis and since the condition " $B > 0$ " in Leaf 2 is matched with one of the conditions defined by the RC policy (i.e., $B > 0 = \text{TRUE}$, the conditions defined by the RC policy are partially matched), Tree 2 will have a TRUE state and thus the correlation engine **111** will cause an operation to occur. Accordingly, the correlation engine **111** deletes Tree 2.

Example 15

In this example, the correlation policy is RNC ($A > 1$ OR $B > 0$), and the correlation engine **111** receives the event stream, having been filtered to remove the C from the event stream A_1CA_2B with event filtering, and thus the event stream received by the correlation engine **111** is event stream A_1A_2B .

Correlation engine **111** scans the event stream A_1A_2B from the first event, A_1 .

Since the conditions are defined by RNC policy, and the first event A_1 is associated with one of the conditions (i.e., $A > 1$), the correlation engine **111** creates a first tree structure, Tree 1. In this example, since the RNC policy defines two conditions (i.e., $A > 1$ and $B > 0$), the correlation engine **111** create two leaves.

Since an event A is detected, the number of A in Leaf 1 is increased into "1", and Leaf 1 of Tree 1 will have a NULL state (since $A=1$ is not $A > 1$).

Tree 1	
Leaf 1	Leaf 2
A = 1 A > 1 = NULL	B = 0 B > 0 = NULL

The correlation engine **111** then scans the next event in the event stream A_1A_2B , A_2 .

The number of A in Leaf 1 will be increased into "2". Accordingly, since "2" is greater than 1, the condition of " $A > 1$ " in Tree 1 will become TRUE.

33

Tree 1	
Leaf 1	Leaf 2
A = 2 A > 1 = TRUE	B = 0 B > 0 = NULL

Since this evaluation is performed on OR-logic basis and since the condition “A>1” in Leaf 1 is matched with one of the conditions defined by the RNC policy (i.e., A>1=TRUE, the conditions defined by the RC policy are partially matched), Tree 1 will have a TRUE state and thus the correlation engine 111 will cause an operation to occur. Accordingly, the correlation engine 111 will delete Tree 1.

After deleting Tree 1, the correlation engine 111 continues to scan the next event in the event stream A₁A₂B, B.

Since event B is associated with one of the conditions (i.e., B>0), the correlation engine 111 creates a second tree structure, Tree 2. In this example, since the RNC policy defines two conditions (i.e., A>1 and B>0), the correlation engine 111 creates two leaves for the second tree.

Since an event B is detected, the number of B in Leaf 2 is increased into “1”.

Tree 2	
Leaf 1	Leaf 2
A = 0 A > 0 = NULL	B = 1 B > 0 = TRUE

Since “1” is greater than 0, the condition of “B>0” in Tree 1 will become TRUE.

Since this evaluation is performed on OR-logic basis and since the condition “B>0” in Leaf 2 is matched with one of the conditions defined by the RNC policy (i.e., B>0=TRUE, the conditions defined by the RNC policy are partially matched), Tree 2 will have a TRUE state and thus the correlation engine 111 will cause an operation to occur. Accordingly, the correlation engine 111 deletes Tree 2.

Examples 16-19 are directed to combination policies. For relaxed-type policies (RC and RNC), since there is no requirement on event order according to the correlation policy, when an event arrives, the correlation engine 111 will check through each node/leaf in a created tree structure to determine whether or not there is any condition related to the event. On the other hand, for strict-type policies (SC and SNC), since there is a requirement on event order, when an event arrives, the correlation engine will first check whether or not the event fulfills the required order, and if it does, the correlation engine 111 will check through the respective node/column in the array structure.

For example, in an RC of (SNC/RNC/SC/RC) combination policy, when an event that is not in the policy occurs, correlation engine will check through the conditions defined by the RC policy and then proceed to the conditions defined by (SNC/RNC/SC/RC) policy(ies). From there, if the correlation engine 111 determines that an event that is not related to any conditions defined in the policy, the correlation engine 111 will determine that the event will never fulfill the contiguous requirement of RC. Accordingly, the correlation engine 111 will delete the tree structure.

In an SC of (SNC/RNC/SC/RC) combination policy, when an event that is not in the policy occurs, correlation engine 111 will check whether or not the event fulfills an event order defined by the SC policy, and the correlation

34

engine 111 will check through the conditions defined by the SC policy and then proceed to the conditions defined by the (SNC/RNC/SC/RC) policy(ies). From there, if the correlation engine 111 determines that an event is not related to any conditions defined in the policy, the correlation engine 111 will determine that the event will never fulfill the contiguous requirement of the SC policy. Accordingly, the correlation engine 111 will delete the array structure.

In an RNC of (SNC/RNC/SC/RC) combination policy, when an event that is not in policy occurs, the correlation engine 111 will check through the conditions defined by the RNC policy and then proceed to the conditions defined by (SNC/RNC/SC/RC) policy(ies). If the parentheses contain RC or SC, then the correlation engine 111 will determine that the event does not fulfill the contiguous requirement of the RC policy or the SC policy, and the correlation engine 111 will delete the tree/array structure. If the parenthesis contains RNC or SNC, then if the correlation engine 111 determines that an event does not relate to any conditions defined by the policy, the correlation engine 111 will ignore the event.

In an SNC of (SNC/RNC/SC/RC) combination policy, when an event that is not in the policy occurs, the correlation engine 111 will check through the conditions defined by the SNC policy and then proceed to the conditions defined by (SNC/RNC/SC/RC) policy(ies), taking into consideration the event order defined by the SNC policy. If the parentheses contain RC or SC, then if the correlation engine 111 determines that the event does not fulfill the contiguous requirement of the RC policy or the SC policy, the correlation engine 111 will delete the tree/array structure. If the parentheses contain RNC or SNC, then the correlation engine 111 will determine that the event does not related to any conditions defined by the policy, and the correlation engine 111 will thus ignore the event.

Examples 16-19 are presented as example scenarios in which the events arrive at the correlation engine 111 without being filtered (i.e., an event(s) not in policy is involved in the NFAT evaluation). It should be noted, however, that the correlation engine 111 is also capable of performing similar operations on filtered events, in which all events involved in the NFAT evaluation are related to the policy, as discussed herein.

Example 16

In this example, the correlation policy is RC (C>0 AND RNC (D>1 OR E>1)), and the correlation engine 111 receives the event stream, C₁FD₁D₂C₂, without filtering.

Correlation engine 111 scans the event stream C₁FD₁D₂C₂ from the first event, C₁.

Since C₁ is related to the conditions defined by the NFAT policy, the correlation engine 111 creates a tree structure with two leaves. The first leaf is a leaf associated with the condition “C>0”, and the second leaf is a leaf associated with the conditions defined by the RNC policy.

In this example, since the RNC policy defines two conditions “D>1” and “E>1”, the correlation engine 111 causes the second leaf to have two sub-leaves, one of which associate with condition “D>1” and the other of which is associated with condition “E>1”.

The number of C in Leaf 1 will be increased to 1. Since the condition “C>0” is fulfilled, the state of Leaf 1 will become TRUE state. However, since the RC policy is an AND-logic based policy, i.e., both condition of “C>0” and conditions defined by RNC policy need to be satisfied, and since the conditions defined by RNC policy have not yet

35

satisfied (i.e., leaf 2 is in a NULL state), the overall state of Tree 1 will be maintained at NULL state.

Tree 1	
Leaf 1	Leaf 2
C = 1 C > 0 = TRUE	
Sub-Leaf 1	Sub-Leaf 2
D = 0 D > 1 = NULL	E = 0 E > 1 = NULL

The correlation engine **111** then scans the second event of the event stream $C_1FD_1D_2C_2$, F, and checks the conditions from the first layer of leaves to the second layer of leaves (i.e., the sub-leaves).

Since F is not associated with any conditions defined by the RC policy and the RNC policy, and the tree structure requires a contiguous condition due to the RC policy, the tree structure Tree 1 will be deleted by the correlation engine **111**.

The correlation engine **111** then scans the next event of the event stream $C_1FD_1D_2C_2$, D_1 .

Since D_1 is related to the conditions defined by the NFAT policy, and since the NFAT policy is a RC policy of a RNC policy, a second tree structure with two leaves will be created, like Tree 1, but because Tree 1 has been deleted, C=0 at this stage of processing the event stream.

The number of D in Leaf 1 of the second layer of leaves (i.e., Sub-Leaf 1) will be increased to 1. Since the condition “D>1” is not yet fulfilled, the state of Leaf 1 of the second layer of leaves will be maintained at NULL state. Subsequently, the overall state of the tree structure will be maintained at NULL state.

Tree 2	
Leaf 1	Leaf 2
C = 0 C > 0 = NULL	
Sub-Leaf 1	
D = 1 D > 1 = NULL	E = 0 E > 1 = NULL

The correlation engine **111** scans the next event of the event stream, D_2 .

The number of D in Leaf 1 of the second layer of leaves will be increased to 2. Since the condition “D>1” is fulfilled, the state of Leaf 1 of the second layer of leaves will become TRUE state. Accordingly, leaf 2 of the first layer of leaves will become TRUE state.

Tree 2	
Leaf 1	Leaf 2
C = 0 C > 0 = NULL	
Sub-Leaf 1	Sub-Leaf 2
D = 2 D > 1 = TRUE	E = 0 E > 1 = NULL

36

However, since the RC policy is an AND-logic based policy, i.e., both states of Leaf 1 of the first layer of leaves and Leaf 2 of the first layer of leaves need to be TRUE in order for the correlation engine **111** to cause an operation to occur, and since the condition “C>0” defined in Leaf 1 of the first layer of leaves has not yet been satisfied (i.e., Leaf 1 of the first layer of leaves is in a NULL state), the overall state of the second tree structure Tree 2 will be maintained at NULL state.

The correlation engine **111** then scans the next event of the event stream, C_2 .

The number of C in Leaf 1 will be increased to 1. Since the condition “C>0” is fulfilled, the state of leaf 1 of the first layer of leaves will become TRUE state.

Tree 2	
Leaf 1	Leaf 2
C = 1 C > 0 = TRUE	
Sub-Leaf 1	Sub-Leaf 2
D = 2 D > 1 = TRUE	E = 0 E > 1 = NULL

Accordingly, the conditions defined by the AND-logic based RC policy will be fulfilled, since both leaves in the first layer of leaves are now having a TRUE state. Accordingly, the tree structure will have a TRUE state, and the correlation engine **111** will cause an operation to occur and delete the second tree structure, Tree 2.

Examples 17-19 demonstrate how the correlation engine **111** handles identical NFAT Policies when an event not in the policy occurs in different orders.

Example 17

In this example, the correlation policy is SNC ($A>0$, $B>0$, $RC(C>1 \text{ OR } D>0)$), and the correlation engine **111** receives the event stream, $AFBC_1C_2$, without filtering.

Correlation engine **111** scans the event stream $AFBC_1C_2$ from the first event, A

Since A is related to the conditions defined by the SNC policy, the correlation engine **111** creates a combination of an array structure with three columns and including a tree structure. The first column is associated with the condition “A>0”, the second column is associated with the condition “B>0”, and the third column is the tree structure which is associated with conditions defined by the RC policy.

The number of A in the first column will be increased to 1, and the first column will have a TRUE state since the condition “A>0” is satisfied. However, since the policy is an AND-logic based policy, i.e., it is required that all three columns to have a TRUE state in order for the overall state of the array structure to become a TRUE state, no operation will be caused to occur by the correlation engine **111** and the overall state of the array structure will be maintained at NULL state.

Array 1			
A = 1	B = 0	Tree 1	
A > 0 = TRUE	B > 0 = NULL	Leaf 1	Leaf 2
		C = 0 C > 1 = NULL	D = 0 D > 1 = NULL

37

The correlation engine **111** then scans the next event of the event stream, F.

Since event F is not an event related to the conditions defined by the policy, and since the correlation engine **111** is still checking the event F for condition “B>0” defined by the SNC policy in the second column and has not yet started the evaluation of RC policy in the third column, due to the non-contiguous characteristic of SNC policy, the correlation engine **111** will ignore event F and continue to scan the next event of the event stream.

The correlation engine **111** then scans the next event of the event stream, B.

The number of B in the second column will be increased to 1, and the second column will have a TRUE state since the condition “B>0” is satisfied. However, since none of the conditions defined by the RC policy has been fulfilled and the state of the third column is NULL, the overall state of the array structure will be maintained at NULL state.

Array 1			
A = 1	B = 1	Tree 1	
A > 0 = TRUE	B > 0 = TRUE	Leaf 1	Leaf 2
		C = 0	D = 0
		C > 1 = NULL	D > 1 = NULL

The correlation engine **111** then scans the next event of the event stream, C₁.

The number of C in leaf 1 of the tree structure will be increased to 1. Since the condition “C>1” is not satisfied, the state of Tree 1 will be maintained at NULL state, and the state of third column of the array structure will be maintained at NULL state. Accordingly, the overall state of the array structure will be maintained at NULL state.

Array 1			
A = 1	B = 1	Tree 1	
A > 0 = TRUE	B > 0 = TRUE	Leaf 1	Leaf 2
		C = 1	D = 0
		C > 1 = NULL	D > 1 = NULL

The correlation engine **111** then scans the next event of the event stream, C₂.

The number of C in Leaf 1 of the tree structure, Tree 1, will be increased to 2. Since the condition “C>1” is satisfied, the state of Tree 1 will become a TRUE state (since the RC policy is an OR-logic based policy and the state of the tree will turn TRUE as long as conditions defined by the RC policy is partially satisfied), and the state of third column of the array structure will become a TRUE state. Accordingly, since all three columns have TRUE state, the overall state of the array structure will become TRUE, and the correlation engine **111** will cause an operation to occur and delete the array structure thereafter.

Example 18

In this example, the correlation policy is SNC (A>0, B>0, RC(C>1 OR D>0)), and the correlation engine **111** receives the event stream, ABFC₁C₂, without filtering.

Correlation engine **111** scans the event stream AFBC₁C₂ from the first event, A.

38

Since A is related to the conditions defined by the SNC policy, the correlation engine **111** creates an array structure with three columns. The first column is associated with the condition “A>0”, the second column is associated with the condition “B>0”, and the third column is associated with conditions defined by the RC policy.

Array 1			
A = 0	B = 0	Tree 1	
A > 0 = NULL	B > 0 = NULL	Leaf 1	
		C = 0	D = 0
		C > 1 = NULL	D > 1 = NULL

The number of A in the first column will be increased to 1, and the first column will have a TRUE state since the condition “A>0” is satisfied. However, since the policy is a strict-type policy, i.e., it is required that all three columns to have a TRUE state in order for the overall state of the array structure to become a TRUE state, no operation will be caused to occur by the correlation engine **111** and the overall state of the array structure will be maintained at NULL state.

Array 1			
A = 1	B = 0	Tree 1	
A > 0 = TRUE	B > 0 = NULL	Leaf 1	Leaf 2
		C = 0	D = 0
		C > 1 = NULL	D > 1 = NULL

The correlation engine **111** then scans the next event of the event stream, B.

The number of B in the second column will be increased to 1, and the second column will have a TRUE state since the condition “B>0” is satisfied. However, since none of the conditions defined by the RC policy has been fulfilled and the state of the third column is NULL, the overall state of the array structure will be maintained at NULL state.

Array 1			
A = 1	B = 1	Tree 1	
A > 0 = TRUE	B > 0 = TRUE	Leaf 1	Leaf 2
		C = 0	D = 0
		C > 1 = NULL	D > 1 = NULL

The correlation engine **111** scans the next event of the event stream, F.

Since event F is not an event related to the conditions defined by the policy, and since the correlation engine **111** has not yet started the evaluation of RC policy in the third column, due to the non-contiguous characteristic of SNC policy, the correlation engine **111** will ignore event F and continue to scan the next event of the event stream.

The correlation engine **111** then scans the next event of the event stream, C₁.

The number of C in Leaf 1 of the tree structure, Tree 1, will be increased to 1. Since the condition “C>1” is not satisfied, the state of Tree 1 will be maintained at NULL state, and the state of the third column of the array structure will be maintained at NULL state. Accordingly, the overall state of the array structure will be maintained at NULL state.

39

Array 1			
A = 1	B = 1	Tree 1	
A > 0 = TRUE	B > 0 = TRUE	Leaf 1	Leaf 2
		C = 1	D = 0
		C > 1 = NULL	D > 1 = NULL

The correlation engine **111** then scans the next event of the event stream, C_2 . The number of C in Leaf 1 of the tree structure will be increased to 2. Since the condition “ $C>1$ ” is satisfied, the state of Tree 1 will become a TRUE state (since the RC policy is an OR-logic based policy and the state of the tree will turn TRUE as long as conditions defined by the RC policy is partially satisfied), and the state of third column of the array structure will become a TRUE state. Accordingly, since all three columns have TRUE state, the overall state of the array structure will become TRUE, and the correlation engine **111** will cause an operation to occur and delete the array structure thereafter.

Example 19

In this example, the correlation policy is SNC ($A>0, B>0, RC(C>1 \text{ OR } D>0)$), and the correlation engine **111** receives the event stream, $ABC_1FC_2C_3$, without filtering.

Correlation engine **111** scans the event stream $ABC_1FC_2C_3$ from the first event, A.

Since A is related to the conditions defined by the SNC policy, an array structure with three columns will be created. The first column is associated with the condition “ $A>0$ ”, the second column is associated with the condition “ $B>0$ ”, and the third column is associated with conditions defined by the RC policy.

Array 1			
A = 0	B = 0	Tree 1	
A > 1 = NULL	B > 0 = NULL	Leaf 1	Leaf 2
		C = 0	D = 0
		C > 1 = NULL	D > 1 = NULL

The number of A in the first column will be increased to 1, and the first column will have a TRUE state since the condition “ $A>0$ ” is satisfied. However, since the policy is a strict-type policy, i.e., it is required that all three columns to have a TRUE state in order for the overall state of the array structure to become a TRUE state, no operation will be caused to occur by the correlation engine **111** and the overall state of the array structure will be maintained at NULL state.

Array 1			
A = 1	B = 0	Tree 1	
A > 0 = TRUE	B > 0 = NULL	Leaf 1	Leaf 2
		C = 0	D = 0
		C > 1 = NULL	D > 1 = NULL

The correlation engine **111** then scans the next event of the event stream, B.

The number of B in the second column will be increased to 1, and the second column will have a TRUE state since the

40

condition “ $B>0$ ” is satisfied. However, since none of the conditions defined by the RC policy has been fulfilled and the state of the third column is NULL, the overall state of the array structure will be maintained at NULL state.

Array 1			
A = 1	B = 1	Tree 1	
A > 0 = TRUE	B > 0 = TRUE	Leaf 1	Leaf 2
		C = 0	D = 0
		C > 1 = NULL	D > 1 = NULL

The correlation engine **111** scans the next event of the event stream, C_1 .

The number of C in Leaf 1 of the tree structure will be increased to 1. Since the condition “ $C>1$ ” is not satisfied, the state of Tree 1 will be maintained at NULL state, and the state of third column of the array structure will be maintained at NULL state. Accordingly, the overall state of the array structure will be maintained at NULL state

Array 1			
A = 1	B = 0	Tree 1	
A > 0 = TRUE	B > 0 = TRUE	Leaf 1	Leaf 2
		C = 1	D = 0
		C > 1 = NULL	D > 1 = NULL

The correlation engine **111** scans the next event of the event stream, F.

Since event F is not an event related to the conditions defined by the RC policy, the correlation engine **111** will delete the tree structure from column three. The conditions of column one and column two of the array structure remain unchanged.

The correlation engine **111** scans the next event of the event stream, C_2 .

Since C is related to the condition “ $C>1$ ” defined by the RC policy, the correlation engine **111** will create a second tree structure (which is similar to the previous tree structure) in column three of the array structure. The number of C in Leaf 1 of the new tree structure will be increased to 1. Since the condition “ $C>1$ ” is not satisfied, the state of Tree 1 will be maintained at NULL state, and the state of third column of the array structure will be maintained at NULL state. Accordingly, the overall state of the array structure will be maintained at NULL state.

Array 1			
A = 1	B = 0	Tree 2	
A > 0 = TRUE	B > 0 = TRUE	Leaf 1	Leaf 2
		C = 1	D = 0
		C > 1 = NULL	D > 1 = NULL

The correlation engine **111** then scans the next event of the event stream, C_3 .

The number of C in Leaf 1 of the tree structure will be increased to 2. Since the condition “ $C>1$ ” is satisfied, the state of Tree 2 will become a TRUE state (since the RC policy is an OR-logic based policy and the state of the tree

will turn TRUE as long as conditions defined by the RC policy is partially satisfied), and the state of third column of the array structure will become a TRUE state. Accordingly, since all three columns have TRUE state, the overall state of the array structure will become TRUE, and the correlation engine 111 will cause an operation to occur and delete the array structure thereafter.

FIG. 2 is a flowchart of a process 200 for managing and/or applying a logic-gate based NFAT structure correlation policy, in accordance with one or more embodiments. In some embodiments, the system 100 (FIG. 1) performs the process 200 and is implemented in or by, for instance, a chip set including a processor and a memory as shown in FIG. 12.

In step 201, event data received from an observability framework is processed to detect a status of a network function. The event data is processed based on two or more conditions defined by a NFAT structure correlation policy having: (1) a policy type comprising a strict contiguous (SC) policy, a strict non-contiguous (SNC) policy, a relaxed contiguous (RC) policy, a relaxed non-contiguous (RNC) policy, or a combination thereof, and (2) a type of logic-gate (for strict-type policies, the type of logic-gate will inherently be AND-logic-gate). The characteristics of each of the SC policy, the SNC policy, the RC policy, and the RNC policy, as well as the processes of evaluating said policies under different conditions and logic-gates, have been described in the above and thus similar descriptions will be omitted herein for simplicity.

In step 203, a first value of a first condition of the two or more conditions is determined.

In step 205, a second value of a second condition of the two or more conditions is determined.

In step 207, the policy type of the correlation policy and a type of logic-gate included in the correlation policy are determined. If the policy type is determined as strict-type policy (e.g., SC policy, SNC policy, or a combination thereof), the type of logic-gate will be determined as AND-logic-gate.

In optional step 209, an order of the two or more conditions is determined based on the policy type.

In step 211, a determination is made as to whether the first condition is satisfied based on a comparison of the first value to a first preset value based on a first operator.

In step 213, a determination is made as to whether the second condition is satisfied based on a comparison of the second value to a second preset value based on a second operator.

In optional step 215, a determination based on the policy type is made regarding an order of the first condition and the second condition among the two or more conditions.

In step 217, a determination is made as to whether the correlation policy is satisfied based on the first value, the second value, the type of logic-gate and the policy type. In some embodiments which optional steps 209 and 215 are included, step 217 comprises a determination as to whether the correlation policy is satisfied based on the first value, the second value, the order of the first condition and the second condition, the type of logic-gate and the policy type. In some embodiments, the event data is processed for a predefined hold time until the correlation policy is satisfied or the predefined hold time is expired. In some embodiments, the event data and the correlation policy is caused to be discarded after the predefined hold time based on a determination that the correlation policy is not satisfied by the end of the predefined hold time. In some embodiments, the event data is caused to be discarded after the predefined hold time based on a determination that the correlation policy is

satisfied by the end of the predefined hold time, and the process returns to step 201 to determine status of the network function based on new event data and the correlation policy for a subsequent predefined hold time after the predefined hold time. In some embodiments, the predefined hold time is a validity time of the determined first value, and if the first value is greater than the first preset value, the event data is further processed to determine whether the second condition is satisfied during the hold time. Then, the first value is caused to be discarded after the predefined hold time based on a determination that the second condition is not satisfied by the end of the predefined hold time.

In some embodiments, based on a determination that the policy type is a SC policy or a SNC policy, the event data is processed until the correlation policy is satisfied. Then, at least a portion of the event data is caused to be discarded based on a determination that the correlation policy cannot be satisfied during the predefined hold time, and/or the predefined hold time is adjusted to facilitate processing the event data until the correlation policy is satisfied.

In some embodiments, based on a determination that the policy type is the SC policy or the SNC policy, the event data is processed until the correlation policy is satisfied. Then, the event data is discarded based on a determination that the correlation policy is satisfied, and the status of the network function is determined based on new event data and the correlation policy after the correlation policy is determined to be satisfied.

In some embodiments, based on a determination that the policy type is the RC policy or the RNC policy, the event data is processed until the correlation policy is satisfied based on a determination of whether the logic-gate included in the correlation policy is one that requires both the first condition and the second condition to be satisfied or one that requires at least one of the first condition or the second condition be satisfied. Then, the event data is discarded based on a determination that the correlation policy is satisfied, and the status of the network function is determined based on new event data and the correlation policy after the correlation policy is determined to be satisfied. In some embodiments, only event data scanned up to a determination that a correlation policy is satisfied or applied toward satisfying the correlation policy is discarded from the event data and a remained of the event data is processed and/or new event data is subsequently processed.

In step 219, an operation is caused to occur based on a determination that the correlation policy is satisfied in step 217. In some embodiments, the operation is caused to occur after the end of the predefined hold time based on a determination that the correlation policy is satisfied during the predefined hold time. In some embodiments, the status of the network function is an operation state of a node in a communication network, and the operation comprises an alarm indicating the network node is inoperative and at least a portion of the communication network is down. In some embodiments, the operation further comprises healing the communication network by fixing or circumventing the network node that is inoperative.

FIG. 3 is a block diagram of a logic-gate based NFAT evaluation 300, in accordance with one or more embodiments.

In NFAT evaluation 300, correlation engine 111 processes event data 301 communicated to correlation engine 111 as discussed with respect to system 100 (FIG. 1). NFAT evaluation 300 is an example demonstrating how correlation engine 111 evaluates received event data 301 which, for simplicity, is discussed with respect to basic example vari-

ables “A,” “B,” “C”. Event data **301** is merely exemplary for discussion purposes and can be any combination of variables, enriched or not enriched, associated with specific event data received by observability framework **101** (FIG. 1).

In NFAT evaluation **300**, event data **301** includes an event stream comprising events A, B, C. Correlation engine **111** processes the event data according to a strict contiguous of strict contiguous correlation policy SC (A, SC (B, C)), which is equivalent to a strict contiguous policy SC (A, B, C). Said policies are stored in second datastore **127**. Effectively, based on the correlation policy in this example, correlation engine **111** is processing the event data **301** to see if an event A arrives before a combination of an event B and an event C, wherein the event B occurs before the event C. The details of the NFAT evaluation (such as the specific conditions defined by the policy, the creation of array, etc.) are omitted for simplicity, and further details will be included in the following examples.

Here, because the correlation policy includes SC-type policies, the logic-gate is inferred as an AND-logic-gate.

Based on the event stream being an event stream comprising events A, B, C, correlation engine **111** determines the strict contiguous correlation policy is satisfied, generates an output **303** that indicates the correlation policy is satisfied, and deletes the event data **301**.

FIG. 4 is a block diagram of a logic-gate based NFAT evaluation **400**, in accordance with one or more embodiments.

In NFAT evaluation **400**, correlation engine **111** processes event data **401** communicated to correlation engine **111** as discussed with respect to system **100** (FIG. 1). NFAT evaluation **400** is an example demonstrating how correlation engine **111** evaluates received event data **401** which, for simplicity, is discussed with respect to basic example variables “A,” “B,” “C”. Event data **401** is merely exemplary for discussion purposes and can be any combination of variables, enriched or not enriched, associated with specific event data received by observability framework **101** (FIG. 1).

In NFAT evaluation **400**, event data **401** includes an event stream comprising events A, B₁, B₂, C₁, C₂. Correlation engine **111** processes the event data according to a strict contiguous of a strict non-contiguous correlation policy, SC (A>0, SNC (B>0, C>0)), that is stored in second datastore **127**. Effectively, based on the correlation policy in this example, correlation engine is processing the event data **401** to see if an event A arrives before a combination of event B and event C, and to see whether one event B or multiple events B occurs before or after one event C or multiple events C.

Here, because the correlation policy includes SC-type and SNC-type policies, the logic-gate is inferred as an AND-logic-gate.

Based on the event stream being A, B₁, B₂, C₁, C₂, correlation engine **111** determines A, B₁, C₁ forms an array that matches the pattern A, B, C defined by the strict contiguous of the strict non-contiguous correlation policy, determines that the strict contiguous of the strict non-contiguous correlation policy is satisfied, generates an output **403** that indicates the correlation policy is satisfied, and deletes the array comprising events A, B₁, C₁.

Given that the correlation policy in this example is a strict contiguous of a strict non-contiguous correlation policy, with event A already being used in the formed array and deleted, the correlation engine **111** determines that events B₂, C₂ cannot possibly satisfy the correlation policy, because

there is no additional event A before B₂ and C₂. Correlation engine **111**, accordingly, deletes events B₂, C₂.

FIG. 5 is a block diagram of a logic-gate based NFAT evaluation **500**, in accordance with one or more embodiments.

In NFAT evaluation **500**, correlation engine **111** processes event data **501** communicated to correlation engine **111** as discussed with respect to system **100** (FIG. 1). NFAT evaluation **500** is an example demonstrating how correlation engine **111** evaluates received event data **501** which, for simplicity, is discussed with respect to basic example variables “A,” “B,” “C”. Event data **501** is merely exemplary for discussion purposes and can be any combination of variables, enriched or not enriched, associated with specific event data received by observability framework **101** (FIG. 1).

In NFAT evaluation **500**, event data **501** includes an event stream comprising events A₁, B₁, B₂, A₂, C₁, C₂. Correlation engine **111** processes the event data according to a strict contiguous of a strict non-contiguous correlation policy, SC (A>0, SNC (B>0, C>0)), that is stored in second datastore **127**. Effectively, based on the correlation policy in this example, correlation engine is processing the event data **501** to see if an event A arrives before a combination of event B and event C, and to see whether one event B or multiple events B occurs before or after one event C or multiple events C.

Here, because the correlation policy includes SC-type and SNC-type policies, the logic-gate is inferred as an AND-logic-gate.

Based on the event stream being A₁, B₁, B₂, A₂, C₁, C₂, correlation engine **111** determines no array that matches the pattern A, B, C defined by the strict contiguous of the strict non-contiguous correlation policy can be formed and determines that the strict contiguous of the strict non-contiguous correlation policy is not satisfied, generates an output **503** that indicates the correlation policy is not satisfied, and deletes an array comprising events A₁, B₁, A₂, because A₂ does not arrive before a combination of event B and event C but instead occurs between an event B and an event C.

Given that the correlation policy in this example is a strict contiguous of a strict non-contiguous correlation policy, with events A₁ and A₂ already being deleted, the correlation engine **111** determines that events B₂, C₁, and C₂ cannot possibly satisfy the correlation policy, because there is no additional event A before event B₂, no events A, B before event C₁, and no events A, B before event C₂. Correlation engine **111**, accordingly, deletes events B₂, C₁, C₂.

FIG. 6 is a block diagram of a logic-gate based NFAT evaluation **600**, in accordance with one or more embodiments. In this embodiment, an event stream similar to the one in FIG. 5 is processed based on a type of correlation policy different from the one in FIG. 5.

In NFAT evaluation **600**, correlation engine **111** processes event data **601** communicated to correlation engine **111** as discussed with respect to system **100** (FIG. 1). NFAT evaluation **600** is an example demonstrating how correlation engine **111** evaluates received event data **601** which, for simplicity, is discussed with respect to basic example variables “A,” “B,” “C”. Event data **601** is merely exemplary for discussion purposes and can be any combination of variables, enriched or not enriched, associated with specific event data received by observability framework **101** (FIG. 1).

In NFAT evaluation **600**, event data **601** includes an event stream comprising events A₁, B₁, B₂, A₂, C₁, C₂. Correlation engine **111** processes the event data according to a strict

non-contiguous of a strict non-contiguous correlation policy, SNC ($A > 0$, SNC ($B > 0$, $C > 0$)), that is stored in second datastore 127. Effectively, based on the correlation policy in this example, correlation engine is processing the event data 601 to see whether one event A or multiple events A arrives before or after a combination of event B and event C, and to see whether one event B or multiple events B occurs before or after one event C or multiple events C.

Here, because the correlation policy includes SNC-type policies, the logic-gate is inferred as an AND-logic-gate.

Based on the event stream being $A_1, B_1, B_2, A_2, C_1, C_2$, correlation engine 111 determines that an array formed by events A_1, B_1, C_1 matches the pattern A, B, C defined by the strict non-contiguous of the strict non-contiguous correlation policy and determines that the correlation policy is satisfied, generates an output 603 that indicates the correlation policy is satisfied, and deletes the array comprising values A_1, B_1, C_1 .

Given that the correlation policy in this example is a strict non-contiguous of a strict non-contiguous correlation policy, with values A_1, B_1, C_1 already being deleted, the correlation engine 111 determines that events B_2, A_2 , and C_2 cannot possibly satisfy the correlation policy, because event A_2 does not arrive before the combination of event B_2 and event C_2 , rather event A_2 is between event B_2 and event C_2 . Correlation engine 111, accordingly, deletes events B_2, A_2 , and C_2 .

FIG. 7 is a block diagram of a logic-gate based NFAT evaluation 700, in accordance with one or more embodiments. In this embodiment, an event stream different from the one in FIG. 6 is processed based on a type of correlation policy similar to the one in FIG. 6.

In NFAT evaluation 700, correlation engine 111 processes event data 701 communicated to correlation engine 111 as discussed with respect to system 100 (FIG. 1). NFAT evaluation 700 is an example demonstrating how correlation engine 111 evaluates received event data 701 which, for simplicity, is discussed with respect to basic example variables "A," "B," "C". Event data 701 is merely exemplary for discussion purposes and can be any combination of variables, enriched or not enriched, associated with specific event data received by observability framework 101 (FIG. 1).

In NFAT evaluation 700, event data 701 includes an event stream comprising events $A_1, B_1, A_2, B_2, C_1, C_2$. Correlation engine 111 processes the event data according to a strict non-contiguous of a strict non-contiguous correlation policy, SNC ($A > 0$, SNC ($B > 0$, $C > 0$)), that is stored in second datastore 127. Effectively, based on the correlation policy in this example, correlation engine 111 is processing the event data 701 to see whether one event A or multiple events A arrives before or after a combination of event B and event C, and to see whether one event B or multiple events B occurs before or after one event C or multiple events C.

Here, because the correlation policy includes SNC-type policies, the logic-gate is inferred as an AND-logic-gate.

Based on the event stream being $A_1, B_1, A_2, B_2, C_1, C_2$, correlation engine 111 determines that a first array formed by events A_1, B_1, C_1 matches the pattern A, B, C defined by the strict non-contiguous of the strict non-contiguous correlation policy and determines the correlation policy is satisfied, generates an output 703 that indicates the correlation policy is satisfied, and deletes the array comprising A_1, B_1, C_1 . Correlation engine 111 also determines that a second array formed by events A_2, B_2, C_2 matches the pattern A, B, C defined by the strict non-contiguous of the strict non-contiguous correlation policy and determines the correlation

policy is again satisfied, generates an output 703 that indicates the correlation policy is satisfied, and deletes the array comprising A_2, B_2, C_2 .

FIG. 8 is a block diagram of a logic-gate based NFAT evaluation 800, in accordance with one or more embodiments.

In NFAT evaluation 800, correlation engine 111 processes event data 801 communicated to correlation engine 111 as discussed with respect to system 100 (FIG. 1). NFAT evaluation 800 is an example demonstrating how correlation engine 111 evaluates received event data 801 which, for simplicity, is discussed with respect to basic example variables "A," "B," "C". Event data 801 is merely exemplary for discussion purposes and can be any combination of variables, enriched or not enriched, associated with specific event data received by observability framework 101 (FIG. 1).

In NFAT evaluation 800, event data 801 includes an event stream comprising events A, B, C. Correlation engine 111 processes the event data according to a relaxed contiguous of a relaxed contiguous correlation policy, RC (A AND RC (B AND C)), that is stored in second datastore 127. Effectively, based on the correlation policy in this example, correlation engine 111 is processing the event data 801 to see if one event A or multiple events A, one event B or multiple events B, and one event C or multiple events C, occur in any order.

Based on the event stream being A, B, C, correlation engine 111 determines that a tree structure formed by A, B, C matches the pattern A, B, C defined by the relaxed contiguous of the relaxed contiguous correlation policy and determines the correlation policy is satisfied, generates an output 803 that indicates the correlation policy is satisfied, and deletes the array comprising values A, B, C.

In this example, correlation engine 111 splits the correlation policy into a tree having a root and leaves, wherein one of the leaves is a sub-tree having a corresponding root and leaves.

FIG. 9 is a block diagram of a logic-gate based NFAT evaluation 900, in accordance with one or more embodiments.

In NFAT evaluation 900, correlation engine 111 processes event data 901 communicated to correlation engine 111 as discussed with respect to system 100 (FIG. 1). NFAT evaluation 900 is an example demonstrating how correlation engine 111 evaluates received event data 901 which, for simplicity, is discussed with respect to basic example variables "A," "B," "C," "X," "Y," "Z". Event data 901 is merely exemplary for discussion purposes and can be any combination of variables, enriched or not enriched, associated with specific event data received by observability framework 101 (FIG. 1).

In NFAT evaluation 900, event data 901 includes an event stream comprising events X, X, Y, Y, C, A, B, Z. Correlation engine 111 processes the event data according to a relaxed contiguous of a strict contiguous correlation policy, RC ($X > 1$ AND $Y > 1$ AND SC ($C > 0$, $A > 0$, $B > 0$) AND $Z > 0$) that is stored in second datastore 127. Effectively, based on the correlation policy in this example, correlation engine 111 is processing the event data 901 to see if event X occurs more than once, event Y occurs more than once, and event Z occurs at least once in any order, with C, A, B occurring in order without another B intervening C and A and without another C intervening A and B.

Here, because the correlation policy includes a portion that is an SC-type policy, the logic-gate in the SC-type policy portion is inferred as an AND-logic-gate.

Based on the event stream being X, X, Y, Y, C, A, B, Z, correlation engine 111 determines that a NFAT structure (which comprises a tree structure and an array structure) formed by events X, X, Y, Y, C, A, B, Z matches the sought after pattern defined by the relaxed contiguous of a strict

contiguous correlation policy and determines the correlation policy is satisfied, generates an output 903 that indicates the correlation policy is satisfied, and deletes the NFAT structure comprising X, X, Y, Y, C, A, B, Z.

Similarly, correlation engine 111 would determine that an event stream C, X, X, A, Y, Y, B, Z would also match the sought after pattern. However, correlation engine 111 would determine that an event stream X, X, A, C, B, Y, Y, Z does not match the sought after pattern (since event A is not between event C and event B, and thus the order defined by the SC policy is not satisfied) and the correlation policy is not satisfied.

FIG. 10 is a block diagram of a logic-gate based NFAT evaluation 1000, in accordance with one or more embodiments.

In NFAT evaluation 1000, correlation engine 111 processes event data 1001 communicated to correlation engine 111 as discussed with respect to system 100 (FIG. 1). NFAT evaluation 1000 is an example demonstrating how correlation engine 111 evaluates received event data 1001 which, for simplicity, is discussed with respect to basic example variables "A," "B," "C," "X," "Y," "Z". Event data 1001 is merely exemplary for discussion purposes and can be any combination of variables, enriched or not enriched, associated with specific event data received by observability framework 101 (FIG. 1).

In NFAT evaluation 1000, event data 1901 includes an event stream comprising events X, Y, X, Y, C, A, C, A, B, B, Z, Z. Correlation engine 111 processes the event data according to a relaxed contiguous of a strict non-contiguous correlation policy, RC ($X>1$ AND $Y>1$ AND SNC ($C>0$, $A>0$, $B>0$) AND $Z>0$) that is stored in second datastore 127. Effectively, based on the correlation policy in this example, correlation engine 111 is processing the event data 1001 to see if events included in data event data 1001 match a pattern X, Y, X, Y, C, A, C, A, B, B, Z based on the relaxed contiguous and the strict non-contiguous policies.

Here, because the correlation policy includes a portion that is an SNC-type policy, the logic-gate in the SNC-type policy portion is inferred as an AND-logic-gate. However, from a process standpoint, this example correlation policy may be considered as being equivalent to RC ($X>1$ AND $Y>1$ AND SNC ($C>0$, $A>0$, $B>0$) AND $Z>0$).

Based on the event stream being X, Y, X, Y, C, A, C, A, B, B, Z, Z, and the relaxed contiguous of a strict non-contiguous correlation policy in this example, correlation engine 111 splits the processing of the event data and thereby creating a first tree structure comprises of multiple leaves. Subsequently, the correlation engine 111 counts the arrival of first event X and first event Y as X_1 , Y_1 , for a first leaf and a second leaf, respectively, and increases the associated count to 1. Then, based on the arrival second event X, and second event Y (e.g., X_2 , Y_2), correlation engine 111 increases the count to 2. Since the X, Y have matched the conditions of " $X>1$ " and " $Y>1$ ", respectively, the state in the first leaf and the second leaf is considered to be TRUE. The conditions of " $X>1$ " and " $Y>1$ " defined by the correlation policy will be satisfied.

When a first event C and first event A arrive, these events are added and stored in array 1 of a third leaf in the first tree structure, and when the second event C and second event A arrive, since the conditions of " $C>0$ " and " $A>0$ " are satisfied

and it is partially matched in the first array 1, as per non-contiguous principles, a second tree structure comprised of a similar quantity of leaves as the first tree will be created. Subsequently, the second event C and second event A are added to a second array, array 2 in the third leaf of the second tree structure. When the first event B arrives, it will go to the first array, array 1 in the third leaf of the first tree structure, since array 1 is created before array 2 and the condition " $B>0$ " in array 1 has not yet matched, and once the strict pattern is matched, a subsequent B event will go to the second array, array 2 in the third leaf of the second tree structure. When a first event Z arrives, it will go to a fourth leaf of the first tree structure since the first tree structure is created before the second tree structure and the condition " $Z>0$ " in the fourth leaf of the first tree structure has not yet matched. When correlation engine 111 determines the pattern in the first tree structure is fully matched with the conditions defined by the correlation policy, the correlation engine 111 deletes the first tree structure and corresponding event data. In this example, since the conditions in the second array, array 2 of the second tree structure is partially matched, the second tree structure is not deleted unless the pattern is matched and the correlation policy is satisfied, or unless a specific hold time has passed. The correlation engine 111 then causes the second tree structure to become the first tree structure and wait for other events to arrive, continuing the evaluation process until the correlation policy is completely satisfied or until a specific hold-time has passed.

The foregoing examples of NFAT evaluation executed by system 100 (FIG. 1) facilitate simultaneous correlation of policies of multiple events or simultaneous correlation of multiple types of policies, thereby enabling the system 100 to monitor the status of a network that involves multiple network functions and complex conditions. The foregoing examples involve the processing and evaluation of NFAT structure correlation policies that involve static conditions, dynamic conditions, or a combination thereof.

FIG. 11 is a sequence diagram of a process 1100 for creating, managing and/or applying one or more of a static or a dynamic NFAT structure correlation policy, in accordance with some embodiments.

System 100 is configured to create and apply NFAT policies based on static and/or dynamic conditions. The above-mentioned Examples 1-19 and embodiments discussed with respect to FIGS. 3-10 are examples of NFAT policies that comprise defined conditions, which may be fully defined by a user or be partially defined by the user and then further defined by the correlation engine 111. Sometimes, a user may or may not know a specific value to which an event should be compared in accordance with an operator when creating a correlation policy to be applied by the correlation engine 111, but the user may know that an event instance should be compared to some other event instance or a variable value based on one or more occurrences of one or more events, or value indicative of an operating state of a network device, network function, etc. Thus, in some embodiments, system 100 makes it possible for user to not specify some or all of the parameters for the policy conditions using fixed values, and to instead input a policy using dynamic conditions that the correlation engine 111, for example, will later determine appropriate information associated with the dynamic conditions and create a NFAT policy having conditions that comprise appropriate values. Namely, in some embodiments, the correlation engine 111 is configured to create a NFAT policy without requiring the user to

specify the contents of the NFAT policy with completely static or fixed values at the time of create.

For correlation policies in which a user has good knowledge of the network, a user may specify the contents of a correlation policy using static conditions by way of user interface 125. But, sometimes when one or more desired policy conditions are complicated, or a variable value provides more flexible or adaptive network management capabilities, or a user simply does not know the exact comparison the correlation policy should make when creating the correlation policy, one or more dynamic conditions may be more appropriate for inclusion in the correlation policy.

Accordingly, in some embodiments, a user does not need to have in-depth knowledge of the network and the specific parameter(s) for the conditions, and can instead simply insert a desired dynamic condition into user interface 125 when creating a correlation policy that the correlation engine 111 can then use to produce a NFAT correlation policy based on the dynamic condition and the associated events, wherein the conditions defined by the NFAT correlation policy are in line with the user's desired conditions.

As a non-limiting example to convey the differences between static conditions and dynamic conditions, assume that a user would like the system to perform an action, or cause an operation to occur, under a simple condition, wherein two events A occur consecutively. In this case, since the parameters for the condition (e.g., $A > 1$) is simple, it is possible for the user to configure a NFAT policy to be, for example, a strict contiguous policy SC ($A > 1$), wherein the condition " $A > 1$ " is static and can be easily determined by the user. Nevertheless, should the user desire the system 100 perform an action under a more complicated condition, for example, when a name of an entity occurs more than or equal to a parameter of another entity, it is difficult for a user to quickly determine the required parameters for configuring the conditions of the NFAT policy, and it is particularly difficult for user that does not have good understanding on the network entities, network status, and some other network information. Accordingly, in some embodiments, system 100 provides capabilities to simplify the process of defining conditions of NFAT policy, without requiring the user to have depth understanding on the network or to have prior knowledge on the required parameters.

In such a case, system 100 makes it possible for a user to input a policy with simple conditions into user interface 125, the policy definition manager 121 then creates a dynamic NFAT policy based on the simple policy, and the correlation engine 111 later processes the conditions of the dynamic NFAT policy to update the condition before using the NFAT policy. Subsequently, the correlation engine 111 will continuously obtain related information from the network events and dynamically set or update the conditions in the dynamic NFAT policy, thereby creating a NFAT policy with condition(s) appropriately defined and capable of being evaluated to determine the status of the network. Since the conditions inputted by the user will be updated by the system, said conditions are called as "dynamic conditions".

The dynamic policy capabilities provided by system 100 simplify the process of defining conditions for NFAT policy, which in turn improves the efficiency of the system 100 and reduces the burden on the user that creates a correlation policy for monitoring the network.

For exemplary purposes to assist in understanding the creation and application of dynamic NFAT policies an "entity" as discussed may be any network element, such as event alarm, pod, network functions, etc. In a condition

" $A > B$ ", both A and B can be considered as an entity, since A and B are representative of any event, function, element, and the like.

In this example, the entity on the left (i.e., entity A in the condition " $A > B$ ") is always defined by some parameter, and thus considered as a static entity. The entity on the right (i.e., entity B in the condition " $A > B$ ") can be a parameter or a dynamic entity. For example, when " $A > 1$ ", the entity on the left is defined by parameter 1, a fixed value. On the other hand, when " $A > B$ ", B is not defined by any known parameter, and thus, B is a dynamic entity, a variable parameter (e.g., value, event, etc.). Conditions which contain a dynamic entity are dynamic conditions, which will be updated by the correlation engine 111 in a later process. Conditions which contain only static entities are static conditions. Each entity has a state, and the NFAT policy, whether static or dynamic, has a state.

In some embodiments, when determining whether a combination of entities and/or conditions are true or false, all entities and/or conditions when processed for determining an overall state of a combination of entities, conditions and/or an NFAT policies must be a TRUE state. For example, if two entities and/or conditions are considered, a TRUE state+TRUE state=TRUE state, TRUE state+FALSE state=FALSE, FALSE state+FALSE state=FALSE.

In some embodiments, alternative options for determining whether a combination of entities and/or conditions are true or false are possible. For example, in some embodiments, if two entities and/or conditions are considered, at least one TRUE state=TRUE. In some embodiments, if two entities and/or conditions are considered, a FALSE state+a FALSE state=TRUE. In some embodiments, if more than two entities and/or conditions are considered, at least one TRUE state=TRUE. In some embodiments, if more than two entities and/or conditions are considered, a greater quantity of TRUE states than FALSE states=TRUE. In some embodiments, if more than two entities and/or conditions are considered, a greater quantity of FALSE states than TRUE states=FALSE. In some embodiments, if more than two entities and/or conditions are considered, an equal quantity of FALSE to TRUE states=FALSE. In some embodiments, if more than two entities and/or conditions are considered, an equal quantity of FALSE to TRUE states=TRUE. In some embodiments, if more than two entities and/or conditions are considered, all FALSE states=FALSE. In some embodiments, if more than two entities and/or conditions are considered, all TRUE states=TRUE. In some embodiments, if two or more entities and/or conditions are considered, some other suitable combination of TRUE/FALSE states are capable of being defined to establish whether the overall combination of entities is TRUE or FALSE.

In the sequence diagram of process 1100, the example entity has a state (identified as "Entity State" in FIG. 11). Generally, a static entity has an Entity State of TRUE, and a dynamic entity has an Entity State of FALSE.

In the example condition of $A > B$, A is a static entity (i.e., Entity State=TRUE) and B is a dynamic entity (i.e., Entity State=FALSE). In the example condition of $A > 1$, A is a static entity (i.e., Entity State=TRUE), while 1 is a parameter and does not have an entity state.

Sequence diagram of process 1100 includes processes involving both an NFAT template and a dynamic NFAT policy having a state (identified as "NFAT State" in FIG. 11). It should be understood that when any one of the entities in the conditions has an Entity States of FALSE, then the NFAT State=FALSE. When all of the Entity_States are TRUE, then the NFAT State=TRUE. It should also be understood that

51

NFAT State=TRUE implies that the process of defining conditions of dynamic NFAT policy is completed, and the dynamic NFAT policy is ready to be used. Additionally, it should be understood that NFAT State=FALSE implies that the process of defining conditions of dynamic NFAT policy is not yet completed, and the dynamic NFAT policy is not yet suitable to be used for determining the status of a network, network entity, network function, network device, etc. A dynamic NFAT policy with NFAT State=TRUE is the generally the same with a normal, static, NFAT policy in which the conditions are well defined (e.g., by the system, by the user, and the like).

The processes included in the sequence diagram of process 1100 are discussed in more detail below.

In step 1101, a user input received by way of user interface 125 for generating a NFAT correlation policy is processed to determine whether the user input indicates one or more of a static condition or a dynamic condition for inclusion in the NFAT correlation policy. A static condition, as discussed, comprises a comparison between a defined entity and a fixed parameter (e.g., a fixed value, a constant, etc.). A dynamic condition, as discussed, comprises a comparison between the defined entity and a variable value. For example, a user inputs one or more condition(s) (e.g., in the form of a simple policy) into user interface 125, the condition(s) input via user interface 125 are transmitted to the policy definition manager 121, and the policy definition manager 121 generates an applicable NFAT element. The applicable NFAT element comprises at least one of an NFAT correlation policy generated based on a determination that the user input indicates a static condition or a NFAT template generated based on a determination that the user input indicates a dynamic condition. For purposes of discussion, in the sequence diagram of process 1100, the user input indicates a dynamic entity base policy, and as such, a dynamic condition. Policy definition manager, therefore, creates an NFAT template in step 1103 based on the condition(s), in which the Entity State=FALSE and NFAT State=FALSE. If, however, the policy definition manager 121 determines that the condition(s) amount to a static condition (i.e., all entities are specified by specific parameters/fixed values), then the policy definition manager 121 creates a normal, static, NFAT policy based on the condition(s).

In either case, the applicable NFAT element is caused to be stored in a memory such as second datastore 127. In some embodiments, the created NFAT policy or NFAT template will be stored in the second datastore 127 and/or first datastore 113, or some other suitable memory.

In step 1105, event data received from a network device associated with a communication network comprising the network device is fetched from inventory database 107 by enricher 105 for processing. In some embodiments, the event data is received directly from the network device. In some embodiments, the event data is received by way of observability framework 101. In some embodiments, when the event data is received directly from the network device, the observability framework 101 is optionally excluded from system 100. In some embodiments, the event data is received directly or indirectly from the network device and stored in inventory database 107. The event data is processed by the enricher 105 based on the applicable NFAT element.

In step 1107, the enricher 105 processes the event data to generate enriched event data. In some embodiments, the enricher 105 searches the first datastore 113 and/or the second datastore 127, or other suitable memory for the

52

applicable NFAT element corresponding to the enriched event data. In some embodiments, the enricher 105 enrich the event data by adding information (e.g., topological information of associated events) to the associated events.

In step 1109, enricher 105 sends the enriched event data to the correlation engine 111.

In step 1111, the correlation engine 111 accesses the second datastore 127 and/or other memory, and determines whether or not the second datastore 127 and/or other memory contains an applicable NFAT element corresponding to the enriched event data. Based on a determination that the applicable NFAT element is stored in the memory, the correlation engine 111 determines whether the applicable NFAT element is an NFAT policy or an NFAT template (i.e., a policy comprising one or more dynamic conditions).

If the correlation engine 111 determines the applicable NFAT element is an NFAT template, the correlation engine 111 fetches the NFAT template from the second datastore 127 and/or the other memory in step 1113, and the process proceeds to step 1115 in which the one or more of correlation engine 111 generates a dynamic NFAT correlation policy, and replaces and/or updates a dynamic parameter (e.g., a variable value) included in the NFAT template with a fixed parameter (e.g., a fixed value) based on the enriched event data.

If the correlation engine 111 determines that the applicable NFAT element is a static NFAT policy, or at least an NFAT policy having all conditions in a true state indicating that the NFAT policy is ready for use, the correlation engine 111 fetches the NFAT policy that is ready for use in step 1117, and the process proceeds to step 1119 in which the correlation engine determines the state of the NFAT policy.

Subsequently, the correlation engine 111 proceeds to step 1119 in which the NFAT state of the NFAT policy or the dynamic NFAT policy is determined. If it is determined that the NFAT state of the NFAT policy is TRUE, which implies that the NFAT policy is a normal, static, NFAT policy and is ready to be used, the process proceeds to step 1121 in which the correlation engine 111 will start to evaluate the received events based on the NFAT policy.

If it is determined in step 1119 that the NFAT state of the NFAT policy is FALSE, which implies that the NFAT policy is a dynamic NFAT policy and not ready to be used, the process proceeds to step 1123 in which the correlation engine 111 determines whether or not the enriched event data contains an event(s) that have information (e.g., topological information) associated with the dynamic entity (i.e., a variable parameter) in the dynamic NFAT policy.

If it is determined in step 1123 that the enriched event data contains event(s) that have information associated with the dynamic entity, then the correlation engine 111 updates the condition of the dynamic NFAT policy in step 1125 based on the information by replacing the variable parameter with an appropriate fixed parameter associated with the information, and then sets the entity state of the respective entity to TRUE in step 1127.

If it is determined in 1123 that the enriched event data does not contain any event(s) that have information associated with the dynamic entity, then the correlation engine 111 will ignore the events and discard the event data in step 1129.

In step 1131, the correlation engine 111 will determine whether or not all Entity States in the dynamic NFAT policy are TRUE. If it is determined that all Entity States are TRUE, then the correlation engine 111 sets the NFAT state of the dynamic NFAT policy to TRUE, and will then return to step 1121 and start to evaluate the events based on the

53

dynamic NFAT policy that is now ready for use. Otherwise, the correlation engine 111 will continuously request enriched event(s) from the enricher 105 and repeat steps 1123-1131 until all Entity States in the dynamic NFAT policy are TRUE and then changes the NFAT state of the dynamic NFAT policy to TRUE, indicating that the dynamic NFAT policy is ready for use in step 1121.

Based on the outcome of the evaluation in step 1121, in which the correlation engine 111 determines the status of the network entity by evaluating the state of the NFAT policy and/or the dynamic NFAT policy, the correlation engine 111 causes an operation to occur according to a determination of whether one or more of the NFAT correlation policy or the dynamic NFAT correlation policy is satisfied.

The following non-limiting Example 20 is provided to demonstrate creation of an NFAT policy having dynamic conditions which, for simplicity, is discussed with respect to basic example variables "A", "B", "C" and "D" for the exemplary correlation policy which, as discussed, may be customized for any particular purpose for identifying whether an operation is to be caused to occur or not. The event data in the following example is merely exemplary for discussion purposes and can be any combination of variables associated with event data.

Example 20

In this example, a user input received by way of user interface 125 indicates dynamic conditions of "A>B" and "C>D", in which events A and C should occur in a pre-defined order, and the correlation engine 111 receives an event stream ABCD.

The policy definition manager 121 determines that the conditions input by the user are dynamic conditions which need to satisfy some order, and thus the policy definition manager 121 creates a strict-type NFAT template, wherein Entity State of the dynamic entity B and dynamic entity D are both FALSE, and wherein the NFAT State of the NFAT template is FALSE. Said NFAT template is then stored in a memory such as one or more of second datastore 127, first datastore 113 and/or some other suitable memory.

The enricher 105 receives events ABCD, retrieves data (e.g., topological information associated with events ABCD) from the inventory database 107, and then processes events ABCD by enriching the appropriate information to one or more of the events ABCD, respectively. Subsequently, the enricher 105 sends enriched events ABCD to the correlation engine 111.

Correlation engine 111 will access the memory and determine whether or not the memory contains an applicable NFAT element. In this example, the memory only contains an NFAT template and does not have a suitable NFAT policy that is ready for use. Accordingly, correlation engine 111 will retrieve the NFAT template from the memory and create a dynamic NFAT policy based on the NFAT template. The conditions of the dynamic NFAT policy are dynamic conditions (i.e., "A>B" and "C>D") that will be updated by the correlation engine 111 before the dynamic NFAT policy is being used to evaluate the enriched events ABCD.

Subsequently, the correlation engine 111 scans through enriched events ABCD, and determines whether or not any of the enriched events ABCD contain information (e.g., topological information) associated with the dynamic entity B or dynamic entity D. In this example, the enriched events A and C do not contain any information associated with dynamic entity B and dynamic entity D, the enriched event

54

B contains information of "B=1", and the enriched event D contains information of "D=2".

The correlation engine 111 then starts scanning the enriched events ABCD from the first event in the stream, i.e., enriched event A. Since the enriched event A does not contain any information associated with dynamic entity B and dynamic entity D, the correlation engine 111 will ignore enriched event data A.

Next, the correlation engine 111 scans the next event in the event stream, i.e., enriched event B. Since the enriched event B contains information associated with dynamic entity B (i.e., B=1), the correlation engine 111 updates the dynamic condition of "A>B" by replacing B with "1" to generate a static condition of "A>1". Accordingly, since "A>1" only has a static entity A, the Entity State of the condition "A>1" is TRUE.

However, because the condition "C>D" is still a dynamic condition in this example, and D is still a dynamic entity having an Entity State of FALSE, the state of the dynamic NFAT policy remains as FALSE.

The correlation engine 111 then scans the next event in the event stream, i.e., enriched event C. Similar to enriched event A, since enriched event C does not contain any information associated with dynamic entity B and dynamic entity D, the correlation engine 111 will ignore enriched event data C.

Subsequently, the correlation engine 111 scans the next event in the stream, i.e., enriched event D. Since the enriched event D contains information associated with dynamic entity D (i.e., D=2), the correlation engine will update the dynamic condition of "C>D" by replacing D with "2" to generate a static condition of "C>2". Accordingly, since "C>2" only has a static entity C, the Entity State of "C>2" is TRUE.

At this point, since all of the entities have an Entity State of TRUE, the correlation engine 111 sets the NFAT state of the dynamic NFAT policy to TRUE.

Subsequently, the correlation engine 111 will evaluate the enriched events ABCD with a NFAT policy having the conditions of (A>1, C>2).

In this example, like in the embodiments discussed above, the process of defining conditions of a NFAT policy can be simplified. The correlation engine 111 and/or the policy definition manager 121 do not require the user to specify the parameters for each of the conditions (e.g., the user does not need to know that the parameters of B and D are "1" and "2", respectively). Instead, the user can simply input dynamic conditions (e.g., conditions comprising dynamic entities/variable values, such as A>B and A>D) into user interface 125 and the correlation engine 111 and/or the policy definition manager 121 will automatically search for information related to the dynamic conditions and update the dynamic conditions to thereby create a NFAT policy with properly defined conditions that is ready for use in evaluating the status of the network.

Accordingly, when creating an NFAT policy that entirely comprises static conditions, entirely comprises dynamic condition, or comprises a combination of static and dynamic conditions, once the NFAT policy is determined to have a TRUE state with respect to determining that the NFAT policy includes static entities and parameters, the final NFAT policy is ready to be used in a manner that corresponds to that discussed with respect to FIGS. 1-10, above.

FIG. 12 is a functional block diagram of a computer or processor-based system 1200 upon which or by which an embodiment is implemented.

55

Processor-based system **1200** is programmed to create, manage and/or apply a correlation policy, as described herein, and includes, for example, bus **1201**, processor **1203**, and memory **1205** components.

In some embodiments, the processor-based system is implemented as a single “system on a chip.” Processor-based system **1200**, or a portion thereof, constitutes a mechanism for performing one or more steps of creating, managing and/or applying a correlation policy.

In some embodiments, the processor-based system **1200** includes a communication mechanism such as bus **1201** for transferring information and/or instructions among the components of the processor-based system **1200**. Processor **1203** is connected to the bus **1201** to obtain instructions for execution and process information stored in, for example, the memory **1205**. In some embodiments, the processor **1203** is also accompanied with one or more specialized components to perform certain processing functions and tasks such as one or more digital signal processors (DSP), or one or more application-specific integrated circuits (ASIC). A DSP typically is configured to process real-world signals (e.g., sound) in real time independently of the processor **1203**. Similarly, an ASIC is configurable to perform specialized functions not easily performed by a more general purpose processor. Other specialized components to aid in performing the functions described herein optionally include one or more field programmable gate arrays (FPGA), one or more controllers, or one or more other special-purpose computer chips.

In one or more embodiments, the processor (or multiple processors) **1203** performs a set of operations on information as specified by a set of instructions stored in memory **1205** related to creating, managing and/or applying a correlation policy. The execution of the instructions causes the processor to perform specified functions.

The processor **1203** and accompanying components are connected to the memory **1205** via the bus **1201**. The memory **1205** includes one or more of dynamic memory (e.g., RAM, magnetic disk, writable optical disk, etc.) and static memory (e.g., ROM, CD-ROM, etc.) for storing executable instructions that when executed perform the steps described herein to create, manage and/or apply a correlation policy. The memory **1205** also stores the data associated with or generated by the execution of the steps.

In one or more embodiments, the memory **1205**, such as a random access memory (RAM) or any other dynamic storage device, stores information including processor instructions for creating, managing and/or applying a correlation policy. Dynamic memory allows information stored therein to be changed. RAM allows a unit of information stored at a location called a memory address to be stored and retrieved independently of information at neighboring addresses. The memory **1205** is also used by the processor **1203** to store temporary values during execution of processor instructions. In various embodiments, the memory **1205** is a read only memory (ROM) or any other static storage device coupled to the bus **1201** for storing static information, including instructions, that is not capable of being changed by processor **1203**. Some memory is composed of volatile storage that loses the information stored thereon when power is lost. In some embodiments, the memory **1205** is a non-volatile (persistent) storage device, such as a magnetic disk, optical disk or flash card, for storing information, including instructions, that persists even when the system **1200** is turned off or otherwise loses power.

The term “computer-readable medium” as used herein refers to any medium that participates in providing infor-

56

mation to processor **1203**, including instructions for execution. Such a medium takes many forms, including, but not limited to computer-readable storage medium (e.g., non-volatile media, volatile media). Non-volatile media includes, for example, optical or magnetic disks. Volatile media include, for example, dynamic memory. Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, a hard disk, a magnetic tape, another magnetic medium, a CD-ROM, CDRW, DVD, another optical medium, punch cards, paper tape, optical mark sheets, another physical medium with patterns of holes or other optically recognizable indicia, a RAM, a PROM, an EPROM, a FLASH-EPROM, an EEPROM, a flash memory, another memory chip or cartridge, or another medium from which a computer can read. The term computer-readable storage medium is used herein to refer to a computer-readable medium.

An aspect of this description is directed to a method comprising processing, by a processor, a user input for generating a non-deterministic finite automata tree (NFAT) correlation policy to determine whether the user input indicates one or more of a static condition or a dynamic condition for inclusion in the NFAT correlation policy. The static condition comprises a comparison between a defined entity and a first fixed parameter, and the dynamic condition comprises a comparison between the defined entity and a variable value. The method also comprises causing an applicable NFAT element to be generated. The applicable NFAT element comprises at least one of the NFAT correlation policy generated based on a determination that the user input indicates the static condition or a NFAT template generated based on a determination that the user input indicates the dynamic condition. The method further comprises causing the applicable NFAT element to be stored in a memory. The method additionally comprises processing event data received from a network device to detect a status of a network entity associated with a communication network comprising the network device. The event data is processed based on the applicable NFAT element. The processing of the event data comprises enriching the event data to generate enriched event data, searching the memory for the applicable NFAT element corresponding to the enriched event data, and based on a determination that the applicable NFAT element is stored in the memory, one or more of: generating a dynamic NFAT correlation policy based on a determination that the applicable NFAT element is the NFAT template, replacing the variable value with a second fixed parameter based on the enriched event data, and determining a state of the dynamic NFAT correlation policy based on the dynamic condition and the enriched event data, or determining the state of the NFAT correlation policy based on a determination that the applicable NFAT element is the NFAT correlation policy. The method also comprises causing an operation to occur based on the status of the network entity according to a determination of whether one or more of the NFAT correlation policy or the dynamic NFAT correlation policy is satisfied.

Another aspect of this description is directed to an apparatus comprising a processor and a memory having instructions stored thereon that, when executed by the processor, cause the apparatus to process a user input for generating a non-deterministic finite automata tree (NFAT) correlation policy to determine whether the user input indicates one or more of a static condition or a dynamic condition for inclusion in the NFAT correlation policy. The static condition comprises a comparison between a defined entity and a first fixed parameter. The dynamic condition comprises a

57

comparison between the defined entity and a variable value. The apparatus is also caused to cause an applicable NFAT element to be generated, the applicable NFAT element comprising at least one of the NFAT correlation policy generated based on a determination that the user input indicates the static condition or a NFAT template generated based on a determination that the user input indicates the dynamic condition. The apparatus is further caused to cause the applicable NFAT element to be stored in a memory. The apparatus is additionally caused to process event data received from a network device to detect a status of a network entity associated with a communication network comprising the network device. The event data is processed based on the applicable NFAT element. The processing of the event data comprises enriching the event data to generate enriched event data, searching the memory for the applicable NFAT element corresponding to the enriched event data, and based on a determination that the applicable NFAT element is stored in the memory, one or more of: generating a dynamic NFAT correlation policy based on a determination that the applicable NFAT element is the NFAT template, replacing the variable value with a second fixed parameter based on the enriched event data, and determining a state of the dynamic NFAT correlation policy based on the dynamic condition and the enriched event data, or determining the state of the NFAT correlation policy based on a determination that the applicable NFAT element is the NFAT correlation policy. The apparatus is also caused to cause an operation to occur based on the status of the network entity according to a determination of whether one or more of the NFAT correlation policy or the dynamic NFAT correlation policy is satisfied.

Another aspect of this description is directed to a non-transitory computer readable medium having instructions stored thereon that, when executed by a processor, cause an apparatus to process a user input for generating a non-deterministic finite automata tree (NFAT) correlation policy to determine whether the user input indicates one or more of a static condition or a dynamic condition for inclusion in the NFAT correlation policy. The static condition comprises a comparison between a defined entity and a first fixed parameter. The dynamic condition comprises a comparison between the defined entity and a variable value. The apparatus is also caused to cause an applicable NFAT element to be generated, the applicable NFAT element comprising at least one of the NFAT correlation policy generated based on a determination that the user input indicates the static condition or a NFAT template generated based on a determination that the user input indicates the dynamic condition. The apparatus is further caused to cause the applicable NFAT element to be stored in a memory. The apparatus is additionally caused to process event data received from a network device to detect a status of a network entity associated with a communication network comprising the network device. The event data is processed based on the applicable NFAT element. The processing of the event data comprises enriching the event data to generate enriched event data, searching the memory for the applicable NFAT element corresponding to the enriched event data, and based on a determination that the applicable NFAT element is stored in the memory, one or more of: generating a dynamic NFAT correlation policy based on a determination that the applicable NFAT element is the NFAT template, replacing the variable value with a second fixed parameter based on the enriched event data, and determining a state of the dynamic NFAT correlation policy based on the dynamic condition and the enriched event data, or determining the

58

state of the NFAT correlation policy based on a determination that the applicable NFAT element is the NFAT correlation policy. The apparatus is also caused to cause an operation to occur based on the status of the network entity according to a determination of whether one or more of the NFAT correlation policy or the dynamic NFAT correlation policy is satisfied.

The foregoing outlines features of several embodiments so that those skilled in the art may better understand the aspects of the present disclosure. Those skilled in the art should appreciate that they may readily use the present disclosure as a basis for designing or modifying other processes and structures for carrying out the same purposes and/or achieving the same advantages of the embodiments introduced herein. Those skilled in the art should also realize that such equivalent constructions do not depart from the spirit and scope of the present disclosure, and that they may make various changes, substitutions, and alterations herein without departing from the spirit and scope of the present disclosure.

What is claimed is:

1. A method, comprising:

processing, by a processor, a user input for generating a non-deterministic finite automata tree (NFAT) correlation policy to determine whether the user input indicates one or more of a static condition or a dynamic condition for inclusion in the NFAT correlation policy, wherein the static condition comprises a comparison between a defined entity and a first fixed parameter, and the dynamic condition comprises a comparison between the defined entity and a variable parameter; causing an applicable NFAT element to be generated, the applicable NFAT element comprising at least one of the NFAT correlation policy generated based on a determination that the user input indicates the static condition or a NFAT template generated based on a determination that the user input indicates the dynamic condition; causing the applicable NFAT element to be stored in a memory; processing event data received from a network device to detect a status of a network entity associated with a communication network comprising the network device, the event data being processed based on the applicable NFAT element, wherein the processing of the event data comprises: enriching the event data to generate enriched event data; searching the memory for the applicable NFAT element corresponding to the enriched event data; and based on a determination that the applicable NFAT element is stored in the memory, one or more of: generating a dynamic NFAT correlation policy based on a determination that the applicable NFAT element is the NFAT template, replacing the variable parameter with a second fixed parameter based on the enriched event data, and determining a state of the dynamic NFAT correlation policy based on the dynamic condition and the enriched event data; or determining the state of the NFAT correlation policy based on a determination that the applicable NFAT element is the NFAT correlation policy; and causing an operation to occur based on the status of the network entity according to a determination of whether one or more of the NFAT correlation policy or the dynamic NFAT correlation policy is satisfied.

59

2. The method of claim 1, wherein the NFAT correlation policy and the dynamic NFAT correlation policy have a corresponding policy type, and the method further comprises:

determining the policy type of the NFAT correlation policy or the dynamic NFAT correlation policy; and
determining whether the NFAT correlation policy or the dynamic NFAT correlation policy is satisfied based on the defined entity, the first fixed parameter or the second fixed parameter, and the policy type.

3. The method of claim 2, wherein determining the policy type comprises determining whether the policy type is a strict contiguous (SC) policy, a strict non-contiguous (SNC) policy, a relaxed contiguous (RC) policy, a relaxed non-contiguous (RNC) policy, or a combination thereof.

4. The method of claim 3, wherein at least one of the NFAT correlation policy or the dynamic NFAT correlation policy comprises two or more of the static or dynamic conditions and, based on a determination that the policy type is at least one of the SC policy of the SNC policy, the method further comprising:

determining an order of the two or more static or dynamic conditions,
wherein the step of determining whether the NFAT correlation policy or the dynamic NFAT correlation policy is satisfied is based on the order of the two or more static or dynamic conditions and the policy type.

5. The method of claim 1, wherein the method further comprises:

processing the event data for a predefined hold time until at least one of the NFAT correlation policy or the dynamic NFAT correlation policy is satisfied or the predefined hold time is expired.

6. The method of claim 5, further comprising:

based on a determination that the at least one of the NFAT correlation policy or the dynamic NFAT correlation policy cannot be satisfied during the predefined hold time, causing at least a portion of the event data to be discarded.

7. The method of claim 5, further comprising:

causing the event data to be discarded after the predefined hold time based on a determination that the at least one of the NFAT structure correlation policy or the dynamic NFAT correlation policy is satisfied by the end of the predefined hold time; and

detecting the status of the network entity based on new event data and the at least one of the NFAT correlation policy or the dynamic NFAT correlation policy for a subsequent predefined hold time after the predefined hold time.

8. The method of claim 5, wherein the operation is caused to occur after the end of the predefined hold time.

9. The method of claim 1, wherein the method further comprises:

processing the event data for a predefined hold time until at least one of the NFAT correlation policy or the dynamic NFAT correlation policy is satisfied or the predefined hold time is expired; and

based on a determination that the at least one of the NFAT correlation policy or the dynamic NFAT correlation policy cannot be satisfied during the predefined hold time, adjusting the predefined hold time to facilitate processing the event data until the at least one of the NFAT correlation policy or the dynamic NFAT correlation policy is satisfied.

60

10. The method of claim 1, wherein the method further comprises:

causing the event data to be discarded based on a determination that at least one of the NFAT correlation policy or the dynamic NFAT correlation policy is satisfied; and

detecting the status of the network entity based on new event data.

11. The method of claim 1, wherein the status of the network entity is an operation state of a node in the communication network, and the operation comprises an alarm indicating the node is inoperative and at least a portion of the communication network is down.

12. The method of claim 11, wherein the operation further comprises healing the communication network by fixing or circumventing the node that is inoperative.

13. An apparatus, comprising:

a processor; and

a memory having instructions stored thereon that, when executed by the processor, cause the apparatus to:

process a user input for generating a non-deterministic finite automata tree (NFAT) correlation policy to determine whether the user input indicates one or more of a static condition or a dynamic condition for inclusion in the NFAT correlation policy, wherein the static condition comprises a comparison between a defined entity and a first fixed parameter, and the dynamic condition comprises a comparison between the defined entity and a variable parameter;

cause an applicable NFAT element to be generated, the applicable NFAT element comprising at least one of the NFAT correlation policy generated based on a determination that the user input indicates the static condition or a NFAT template generated based on a determination that the user input indicates the dynamic condition;

cause the applicable NFAT element to be stored in a memory;

process event data received from a network device to detect a status of a network entity associated with a communication network comprising the network device, the event data being processed based on the applicable NFAT element, wherein processing the event data comprises:

enriching the event data to generate enriched event data;

searching the memory for the applicable NFAT element corresponding to the enriched event data; and

based on a determination that the applicable NFAT element is stored in the memory, one or more of:

generating a dynamic NFAT correlation policy based on a determination that the applicable NFAT element is the NFAT template, replacing the variable parameter with a second fixed parameter based on the enriched event data, and determining a state of the dynamic NFAT correlation policy based on the dynamic condition and the enriched event data; or
determining the state of the NFAT correlation policy based on a determination that the applicable NFAT element is the NFAT correlation policy; and

cause an operation to occur based on the status of the network entity according to a determination of whether one or more of the NFAT correlation policy or the dynamic NFAT correlation policy is satisfied.

61

14. The apparatus of claim 13, wherein the NFAT correlation policy and the dynamic NFAT correlation policy have a corresponding policy type, and the apparatus is further caused to:

determine the policy type of the NFAT correlation policy or the dynamic NFAT correlation policy; and

determine whether the NFAT correlation policy or the dynamic NFAT correlation policy is satisfied based on the defined entity, the first fixed parameter or the second fixed parameter, and the policy type.

15. The apparatus of claim 14, wherein the policy type comprises a strict contiguous (SC) policy, a strict non-contiguous (SNC) policy, a relaxed contiguous (RC) policy, a relaxed non-contiguous (RNC) policy, or a combination thereof.

16. The apparatus of claim 15, wherein at least one of the NFAT correlation policy or the dynamic NFAT correlation policy comprises two or more of the static or dynamic conditions and, based on a determination that the policy type is at least one of the SC policy of the SNC policy, the apparatus is further caused to:

determine an order of the two or more static or dynamic conditions,

wherein the determination of whether the NFAT correlation policy or the dynamic NFAT correlation policy is satisfied is based on the order of the two or more static or dynamic conditions and the policy type.

17. The apparatus of claim 13, wherein the apparatus is further caused to:

process the event data for a predefined hold time until at least one of the NFAT correlation policy or the dynamic NFAT correlation policy is satisfied or the predefined hold time is expired.

18. The apparatus of claim 17, wherein the apparatus is further caused to:

cause at least a portion of the event data to be discarded based on a determination that the at least one of the NFAT correlation policy or the dynamic NFAT correlation policy cannot be satisfied during the predefined hold time.

19. The apparatus of claim 18, wherein the apparatus is further caused to:

cause the event data to be discarded after the predefined hold time based on a determination that the at least one of the NFAT structure correlation policy or the dynamic NFAT correlation policy is satisfied by the end of the predefined hold time; and

detect the status of the network entity based on new event data and the at least one of the NFAT correlation policy

62

or the dynamic NFAT correlation policy for a subsequent predefined hold time after the predefined hold time.

20. A non-transitory computer readable storage medium having instructions stored thereon that, when executed by a processor, cause an apparatus to:

process a user input for generating a non-deterministic finite automata tree (NFAT) correlation policy to determine whether the user input indicates one or more of a static condition or a dynamic condition for inclusion in the NFAT correlation policy, wherein the static condition comprises a comparison between a defined entity and a first fixed parameter, and the dynamic condition comprises a comparison between the defined entity and a variable parameter;

cause an applicable NFAT element to be generated, the applicable NFAT element comprising at least one of the NFAT correlation policy generated based on a determination that the user input indicates the static condition or a NFAT template generated based on a determination that the user input indicates the dynamic condition;

cause the applicable NFAT element to be stored in a memory;

process event data received from a network device to detect a status of a network entity associated with a communication network comprising the network device, the event data being processed based on the applicable NFAT element, wherein processing the event data comprises:

enriching the event data to generate enriched event data;

searching the memory for the applicable NFAT element corresponding to the enriched event data; and

based on a determination that the applicable NFAT element is stored in the memory, one or more of:

generating a dynamic NFAT correlation policy based on a determination that the applicable NFAT element is the NFAT template, replacing the variable parameter with a second fixed parameter based on the enriched event data, and determining a state of the dynamic NFAT correlation policy based on the dynamic condition and the enriched event data; or determining the state of the NFAT correlation policy based on a determination that the applicable NFAT element is the NFAT correlation policy; and

cause an operation to occur based on the status of the network entity according to a determination of whether one or more of the NFAT correlation policy or the dynamic NFAT correlation policy is satisfied.

* * * * *